# Embedding and Extracting Domain Knowledge in Machine Learning for Predictive Maintenance

Divish Rengasamy

August 8, 2022

# Acknowledgements

Firstly, I would like to express my deepest gratitude to my supervisors, Dr Grazziela Figueredo and Dr Benjamin Rothwell. Without their endless support and guidance this thesis would not be completed. I will forever cherish the memories of us chatting, laughing, debating, and having weekly supervision meetings in Starbucks.

Secondly, I would like to thank the University of Nottingham and the Marie Skłodowska-Curie Actions programme for funding my PhD. Furthermore, I would like to give thanks to the Institute for Aerospace Technology, Dr Helen Thomas, Hadrian Moran, Dr Tao Yang, and Prof Serhiy Bozhko for their endless support.

I would like to thank the friends that I've made along the way, Jimi, Maryam, Mina, Neeshe, Kwabena, Golnar, Chen, Tajul, Direnc, and many more for the countless memories. Special thanks to Sarah, Daya, Anna, Fereidoon, and Aaron for being a ray of sunshine on cloudy days and making the ordinary moments in life extraordinary.

With the utmost love, I want to thank my parents, sisters, and brothers. Without the inspiration, drive, care, and support that they have given me, I won't be the person I am today.

# Abstract

As sensors become ubiquitous in condition-based maintenance (CBM) for the aerospace sector, there is a significant increase in data available to diagnose and prognose aerospace components' health and longevity. Machine Learning (ML), and more specifically, Deep Learning (DL), are popular tools deployed to analyse big data for CBM. For safety-critical systems such as aerospace, the ML models' predictive power to accurately diagnose fault and prediction of Remaining Useful Life (RUL) is crucial. Furthermore, the ability to explain and justify ML models' output is essential in decision making for CBM. In this thesis, DL models' predictive capability is improved by developing novel approaches that embed knowledge implicitly and explicitly into the loss function for condition-based maintenance. A proposed regression-based weighted loss functions that implicitly embeds knowledge allows the DL model to learn and focus on hard-to-learn instances, while minimising worst-case predictions and providing improved predictive performance. The proposed regression-based weighted loss function, along with an existing classification-based weighted loss function, namely, Focal Loss, are compared to traditional loss functions using aerospace gas turbine engines' RUL prediction and fault detection of the air pressure system, respectively.

Furthermore, an asymmetric loss function that biases the DL models to favour early predictions in RUL is proposed. The asymmetric loss functions is tested on the same aerospace gas turbine engines dataset. The weighted loss functions and asymmetric loss function is able to increase the predictions accuracy over traditional loss functions. In addition to embedding knowledge, this thesis also introduces a new framework to extract knowledge from a learned model to explain the ML model's outputs by identifying important features driving the predictions. We proposed a new ensemble feature importance framework that fuses multiple ML models and their feature importance calculation approaches using both crisp and fuzzy decision fusion to create a more accurate and interpretable post-hoc explanation for the ML models. Subsequently, a fuzzy ensemble feature importance (FEFI) framework is proposed to overcome the shortcomings of crisp value-based ensemble feature importance. Both the new crisp and fuzzy frameworks are investigated using synthetic data under different conditions and also using a real-world case study of factors that affect creep rate in additive manufactured materials. Experiment results reveal that for our case studies FEFI framework is qualitatively more accurate in determining the feature importance compared to the crip valued-based ensemble feature importance framework and also traditional approaches.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

**45M** 45 degree Multi Laser

**45S** 45 degree Single Laser

**AdaBoost** Adaptive Boosting

**AE** Absolute Error

**AFT** Accelerated Failure Time

**AM** Additive Manufacturing

**ANN** Artificial Neural Network

**APS** Air Pressure System

**AUC** Area Under the Curve

**CBM** Condition-based Maintenance

**CE** Cross-Entropy

**CMAPSS** Commercial Modular Aero-Propulsion System Simulation

**CNN** Convolutional neural networks

**CPH** Cox Proportional Hazard

**DL** Deep Learning

**DNN** Deep Neural Network

**DT** Decision Tree

**ERM** Empirical Risk Minimisation

**ET** Extra Trees

**FEFI** Fuzzy Ensemble Feature Importance

**FI** Feature Importance

**FIS** Fuzzy Inference System

**FL** Focal Loss

**FNR** False Negative Rate

**FOR** False Omission Rate

**GAM** Generalised Additive Models

**GBT** Gradient Boosted Trees

**GE** General Electric

**GRU** Gated Recurrent Unit

**HM** Horizontal Multi Laser

**HPC** High Pressure Compressor

**HS** Horizontal Single Laser

**KL** Kullback-Leibler

**KNN** K-Nearest Neighbour

**LASSO** Least Absolute Shrinkage and Selection Operator

**LIME** Local interpretable model-agnostic explanations

**LOCO** Leave-One-Case-Out

**LPBF** Laser Powder Bed Fusion

**LSTM** Long Short-Term Memory

**MAD** Median Absolute Deviation

**MAE** Mean Absolute Error

**MC** Marginal Contribution

**MDI** Mean Decrease Impurity

**MF** Membership Functions

**ML** Machine Learning

**MLP** Multi-Layer Perceptron

**MME** Multi-Method Ensemble

**MSE** Mean Squared Error

**OOB** Out-Of-Bag

**PAC** Probably Approximately Correct

**PDP** Partial Dependence Plot

**PI** Permutation Importance

**PR** Precision-Recall

**RATE** RAnk correlation with majority voTE

**RF** Random Forest

**RMSE** Root Mean Squared Error

**RNN** Recurrent Neural Network

**ROC** Receiver Operating Characteristic

**RUL** Remaining Useful Life

**SGD** Stochastic Gradient Descent

**SHAP** SHapley Additive exPlanations

**SMOTE** Synthetic Minority Over-Sampling Technique

**SVM** Support Vector Machine

**SVR** Support Vector Regression

**VM** Vertical Multi Laser

**VS** Vertical Single laser Stripe

**VSM** Vertical Single laser Meander

# Chapter 1

# Introduction

Many industries, such as airlines, rely heavily on the availability and reliability of their assets. The likelihood of aeroplane parts becoming faulty or reaching their end-of-life increases as they are used, or they fail early in their life due to manufacturing errors. Their deterioration poses a risk to the airlines, passengers and the wider society. Additionally, the cost of maintaining the health of this equipment is high. According to the UK's Department for Business Innovation & Skills, maintaining aircraft accounts for 13% of the total airlines operating costs, which is only 25% lower than the figures paid for fuel [1]. Failure to provide adequate and timely maintenance can lead to fatalities, monetary loss, and intangible losses, such as impact on reputation and on the environment. A common way to manage safety risks is to perform scheduled maintenance. While scheduled maintenance is capable of ensuring assets operate optimally, it is cost- and labour-intensive [2].

In addition, time-based maintenance relies on expert knowledge of aircraft main-

tainers to make immediate corrections if fault occurs. Furthermore, it does not collect or utilise aircraft conditions' data to aid in predicting the health status and longevity of a particular aircraft. As an alternative to address some of these limitations, condition-based maintenance (CBM) has become one of the maintenance strategies to minimise the cost and labour necessary to ensure assets' availability and reliability. CBM, and more specifically predictive maintenance collects data by using sensors to create predictive models. Compared to scheduled maintenance and corrective maintenance, predictive maintenance is more cost-effective due to reduction in labour and downtime of assets [3]. Predictive maintenance models aim at forecasting the future state of the assets and recommend the appropriate maintenance approach, guided by inferences performed based on the data collected. For example, according to General Electric (GE), each of its engine produces approximately 1 Terabyte of data from hundreds of sensors for each flight [4] – all of which can be used to enhance maintenance plans and improve the operations of aircraft. Predictive maintenance often uses Machine Learning (ML), and more specifically Deep Learning (DL) methods to execute prediction tasks, and it has been successful in many areas, such as in manufacturing and aerospace [5, 6, 7].

While ML in predictive maintenance has been successful, there are scenarios where the ML models can fail in unexpected ways that result in undesirable consequences. Some instances of such failures are found in several safety-critical areas, such as healthcare and transportation. For example, in 2016, IBM's "Watson for Oncology" ML product was found to be making erroneous cancer treatment advice, causing the product to halt for further usage [8]. In 2019, a Tesla vehicle with ML

autopilot system activated failed to detect an oncoming truck, resulting in driver fatality [9]. With the shortcomings of ML in mind, the European Union Aviation Safety Agency and the European commission's high-level expert group in artificial intelligence issued reports defining the essential requirements for safe and trustworthy Neural Networks. Two shared requirements between the reports are Accuracy — **correctness** of ML system output; and **explainability** — justification of ML systems ' output.

To the best of our knowledge, the lack of domain knowledge incorporation and extraction with DL models in predictive maintenance for related areas to aerospace and transport is a source of problem that leads to decreased predictive performance and difficulty explaining models' output. For example, a common task in predictive maintenance in aerospace is to determine the Remaining Useful Life (RUL) of gas turbine engine. Without domain knowledge, early or late prediction by the same unit of time are viewed as equally wrong; for example, if predictive model A and B predicts the RUL of machine to be 4 and 6 months, respectively, and the actual RUL is 5 months, the two predictive models have an error of 1 month. However, it is safer to predict RUL earlier so that intervention can occur before failure. This is an example of a domain knowledge to be embedded in DL models to produce more accurate, thus, higher correctness of ML output while being more reliable and safe in RUL predictions.

In addition to improving prediction accuracy in predictive maintenance tasks, the ability to accurately explain the output of predictive models is crucial to understand model behaviour and outputs. A common explainability process is to identify the

most important features that contribute to the model output. In ML, a feature is an individual measurable property or characteristic of a phenomenon (an independent variable), for example, the speed of turbine is a feature of gas turbine engine. The measurement of the impact of features on predictive model's outcomes is known as Feature Importance (FI). However, there is a lack of consensus of the current approaches in determining the importance of feature attribution for ML decision making and it poses a problem for safety-critical systems, as the explanation offered for the outcomes obtained is likely to be unreliable. There is the need for more reliable and accurate ways of establishing FI. Therefore, there is an increased interest in improving the ML model explainability to improve both predictive capability and interpretation of the outputs, especially for safety-critical applications. **Therefore, this thesis focuses on improving predictive capability of ML methods in predictive maintenance by including domain knowledge within the ML pipeline to improve prediction accuracy and enhancing the model's prediction explainability.**

## 1.1 Condition-based Maintenance

One of the earliest recorded implementations of Condition-based Maintenance (CBM) is by a british scientist named Conrad Hal (C.H.) Waddington [10]. While working for the Royal Air Force during the World War II, C.H. Waddington observed that the number of unscheduled maintenance for aircraft goes up after it underwent their regular scheduled maintenance. He found out that short-cycle scheduled maintenance

disturbed the satisfactory state of aircraft, contributing to an increase in breakdown. To solve the issue, Waddington's team proposed that aircraft should only be maintained if they achieved certain conditions indicating deterioration. Figure 1.1 illustrates the difference between scheduled, corrective, and predictive maintenance.

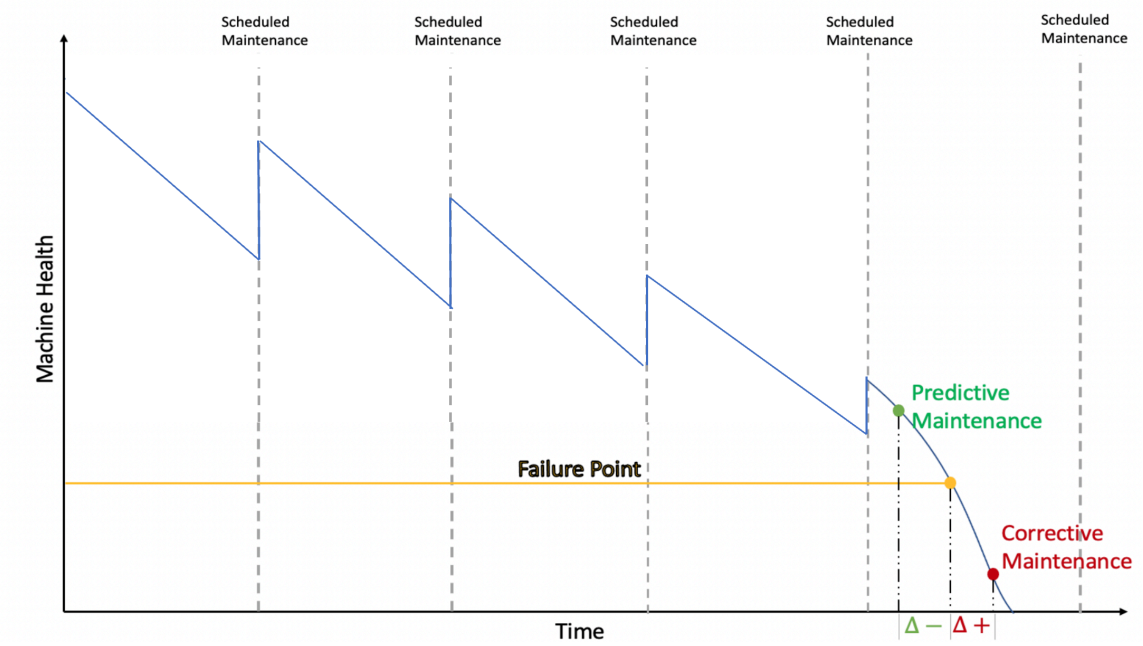



Figure 1.1: A comparison between scheduled, corrective, and predictive maintenance. The grey dotted vertical lines represents the scheduled maintenance overtime. The yellow lines indicates the point the target machines is at the point of failure. Predictive maintenance aims to intervene at the point of failure of earlier ($\Delta-$), while corrective maintenance intervenes at the point of failure or later ($\Delta+$).

A common way of conducting predictive maintenance are model-based and data-driven based predictive systems [11, 12, 7, 6]. Model-based and data-driven based

approaches to predictive maintenance are often used in parallel and in synergy for different system complexities. Model-based approaches model the functional relations between individual components within the asset's system and it is heavily reliant on domain knowledge. The relationship between the individual components can be seen as the cause-effect relationship to support the prediction of deterioration. Therefore, model-based approaches are best used when the systems being modelled are less complex for predictive maintenance to be implemented [13] or using simple/abstract models to represent complex systems. Examples of model-based predictive maintenance include but not limited to, batteries' state of health prediction [14], modelling the aircraft control system to detect actuator failures [15], detecting sensors failures in aircraft control system kinematic models [16], locating faults in power distribution systems using load models [17], and cryogenic propellant loading system diagnosis in spacecraft using physics model that describes the complex dynamics of liquid hydrogen filling [18]. The model-based predictive maintenance approaches mentioned here employ similar strategy. First, a digital model of the system is created in software. Second, gather the data of the system and input it into simulation model to predict the state of the system. In the batteries' state of health prediction example, the authors combined physics-based modelling of Li-ion battery and sequential design of simulation experiments to create an accurate state of health estimator. The model was created in the COMSOL multiphysics software to predict the state of health of the batteries given its condition. Despite its disadvantage, model-based approaches have been successfully implemented for a long time and will continue to be deployed, as they are very useful and reliable for small subsystems CBM implementations.

Additionally, those approaches can be implemented alongside signal-based and data-driven methods for both simple and complex systems. Signal-based approaches refer to monitoring systems based on time-frequency analysis on data collected to detect faults [19]. All three approaches, signal, model, and data-driven are typically implemented together (or, sometimes, using a hybrid version [20]) to perform diagnosis and prognosis of different complexity levels within an aerospace, as shown in Figure 1.2:

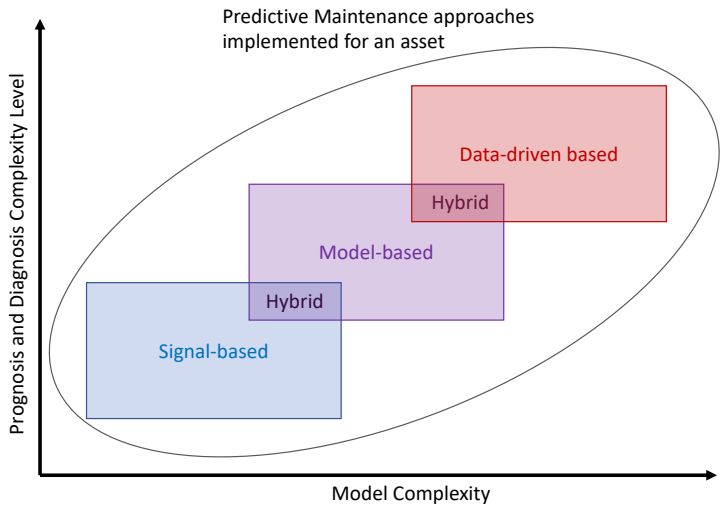



Figure 1.2: Assets typically utilise all three different predictive maintenance approaches, namely, signal, model, and data-driven based. Signal-based approach is commonly used for simple parameter out-of-range detection, while the model-based approach is reserved for a smaller subsystem, and a data-driven approach for larger or even the whole asset's system is difficult to model.

Generally, data-driven predictive maintenance requires far less domain knowledge to function. Domain knowledge is the specialised knowledge related to the area that predictive maintenance is being applied. For example, early prediction is better than

late prediction in RUL task is a domain knowledge within the field of reliabiltiy engineering. It relies on the predictive model to identify patterns regarding the health of assets in the dataset. The complexity of predictive models ranges from statistical models with a couple of parameters to deep learning models [21], with millions of parameters. Weibull distribution, Cox Proportional Hazard (CPH) model, and Accelerated Failure Time (AFT) models are statistical tools in predicting RUL [22, 23, 24]. The Weibull distribution uses time-to-failure data to learn the distribution parameters for prediction. CPH is a semi-parametric regression that uses a hazard function to predict RUL. The hazard function describes the risk of an event happening over time. The primary assumption of CPH is the hazard rates remain constant over time, which is not often the case with the deterioration of machines. Therefore, the assumption of CPH may lead to potential misinterpretation of results [25]. A useful alternative to CPH for modelling RUL is AFT. AFT models event with non-proportional hazard as it uses Weibull or log-normal distribution in its underlying regression model for RUL predictions. While statistical modelling is useful when there are only a few, low-dimensional data points, it faces difficulty when the data are large and high-dimensional as it becomes computationally intractable, leading to the adoption of ML methods.

## 1.2 Machine Learning in Predictive Maintenance

The use of ML is another common approach in creating models for predictive maintenance. ML methods have found to be more accurate in predicting time-to-event

data compared to statistical models, especially when large data is present [26].

A common learning paradigm for ML methods is supervised learning where the models learn mainly from labelled data. The predictive model is obtained by iteratively adjusting its parameters based on the training error, which is obtained by a loss function that calculates the difference between predicted output and the actual output. Loss function is an important component of supervised learning, as different loss functions may result in different model learning. As loss function acts as a feedback loop to the model on how to adjusts its parameter to produce more accurate predictions, changing the loss functions can potentially affect the way model learns. To the best of our knowledge, current implementation of loss function in ML for predictive maintenance are not domain specific, but rather a general version is applied to different ML and DL models.

A common supervised ML algorithm is Support Vector Machine (SVM). SVM maps the data from a low to high dimensional input space and subsequently creates a maximum-margin hyperplane to classify the data [27]. Support Vector Regression (SVR) is an extension to SVM that performs regression [28]. SVM and SVR have been applied extensively to many applications of CBM such as, aerospace engine RUL prediction [29], estimating decay rate in the naval propulsion plant [30], diagnosing electrical failures in induction motors [31], and predicting the RUL of a high pressure liquefied natural gas pump [32].

Another class of supervised learning algorithms are tree-based models, such as Decision Tree (DT), Random Forest (RF) and Gradient Boosted Trees (GBT). DT is a tree-like learning model that chains one to several decision rule based on knowl-

edge learned in training data. RF is an ensemble of DTs, and each DT produces an output [33]. The final output of the RF is based on the majority vote of all DTs. RF trains a new DT in each iteration through bagging. Bagging is a learning approach where a random subset of the training data is selected to train each DT. GBT works similarly to RF as it is also an ensemble of DT. GBT differs from RF in the way the DTs are ensembled. GBT grows and improves upon the existing DTs sequentially [34]. GBT grows iteratively by continuously adding more DTs to the final model and checking error at each iteration until the desired accuracy has been met.RF and GBT have been applied widely to different domains, such as aircraft engine fault diagnosis [35] and prognosis [36], fault diagnosis in gearbox [37], maintenance of railway switches [38], and fault classification in high-voltage current transformers [39]. The literature on the maintenance of railway switches and fault classification in high-voltage current transformer by Bukhsh *et al.* and Khalyasmaa *et al.*, respectively, utilises FI estimates as a post-inference model interpretation tool. The FI extracted in the case of Bukhsh *et al.* is drop-FI calculation whereby their tree-based models are trained n-times where n is the number of features. A feature is excluded for each round of training and the accuracy is recorded. The FI is reflected by the changes of magnitude in accuracy relative to the model trained with all features. For Khalyasmaa *et al.*, the FI obtained from their tree-based models — RF and GBT is based on gini impurity [40]. Gini impurity calculates the FI as the sum of number of splits across all trees that include the feature. The importance increases with the number of splits that include a particular feature. While the FI methods adopted by Bukhsh *et al.* and Khalyasmaa *et al.* are widely accepted and

commonly used, they have downsides such as the possibility of FI methods being unreliable due to the calculations being based on a single ML model and single FI method, and therefore the idiosyncrasies of a particular model when learning possible data patterns. Furthermore, there are issues such as uncertainties of the FI and non-intuitive representation of FI, both of which is important during decision-making especially in safety-critical applications. The wide availability of ML algorithms and diversity of FI techniques complicates the selection of models, the FI approaches to be used, reliable interpretation of models from a given ML method and FI approach, and data representativeness. Different ML models may generate different FI values due to variations in their learning algorithms. Similarly, different FI techniques may produce different importances with the same ML algorithms or training data. For example, models that adequately map the independent to dependent variable mappings using linear functions will generate unambiguous FIs while for significantly nonlinear models, FI is usually a local, context-dependent property of the response surface. To add to the complication, some FI methods are model-agnostic, while others are model-specific.

In addition to traditional ML methods, DL models are an extension of Artificial Neural Network (ANN)s to larger architecture. Deep Neural Network (DNN) is structurally similar to the neural network, where there are three types of layers, i.e., input, hidden, and output layers. A DNN can have any arbitrary number of hidden layers greater than one, with each layer consisting of one or more nodes. The hidden layers are where most of the network's parameters — weights and biases are located. The parameters of the network are most commonly optimised using

the Backpropagation algorithm [41] during training. DL models are well-suited for discovering useful knowledge in high-dimensional data [42]. Therefore, there has been a sharp increased in utilising DL for predictive maintenance research in recent years, as Figure 1.3 illustrates.

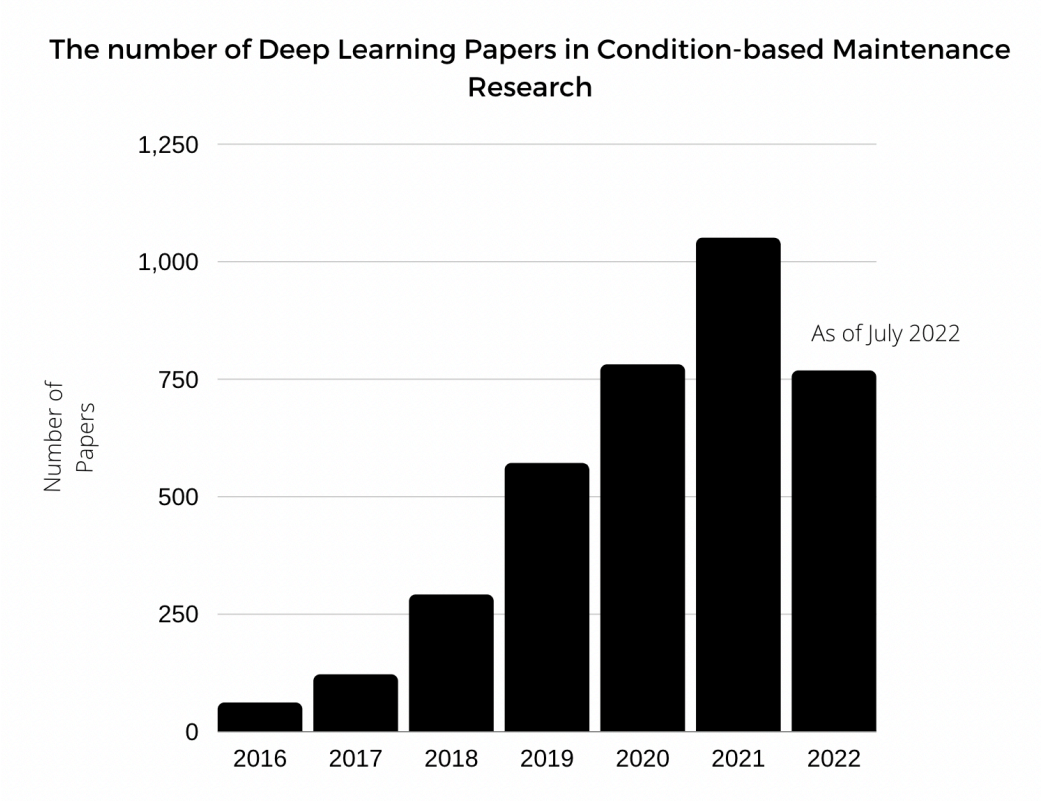



Figure 1.3: The number of Deep Learning papers in predictive maintenance research since 2016 according to Google Scholar.

Our search in the literature has revealed that most advances in the area of using ML and DL in predictive maintenance involve changing and/or improving the architecture employed. While different ML and DL architectures have been widely adopted in predictive maintenance [7], the other components of the ML prediction

pipeline are rarely discussed. Those include the utilisation of non-domain specific loss function to improve prediction accuracy and the lack of explainable outputs. In the next section, the identified shortcomings and research gaps from related research are further discussed.

## 1.3 Research Gaps of Machine Learning in Predictive Maintenance

As previously mentioned, the two main requirements for learning-based models in predictive maintenance are the accuracy and explainability of their predictions. Most advances in CBM and predictive maintenance that utilise DL focus on changing the architecture of the DL models, and therefore there is a gap with regards to improving the proficiency of DL through other aspects of DL learning and training pipelines. The accuracy of DL outputs is determined by several primary factors, namely, data quality [43], model architecture, loss functions, regulariser [44], optimiser [45], and knowledge embedding [46] as shown Figure 1.4.

Regarding accuracy, the focus of this thesis is to modify the loss functions calculations to improve DL predictions in the context of predictive maintenance. Our hypothesis is that the inclusion of domain-specific knowledge in the loss function will lead to improved predictions for predictive maintenance.

Another important research gap in predictive maintenance research is that there is increasing but limited usage of explainability methods for complex ML models [47, 48]. However, research shows that explainability methods' outputs should not be

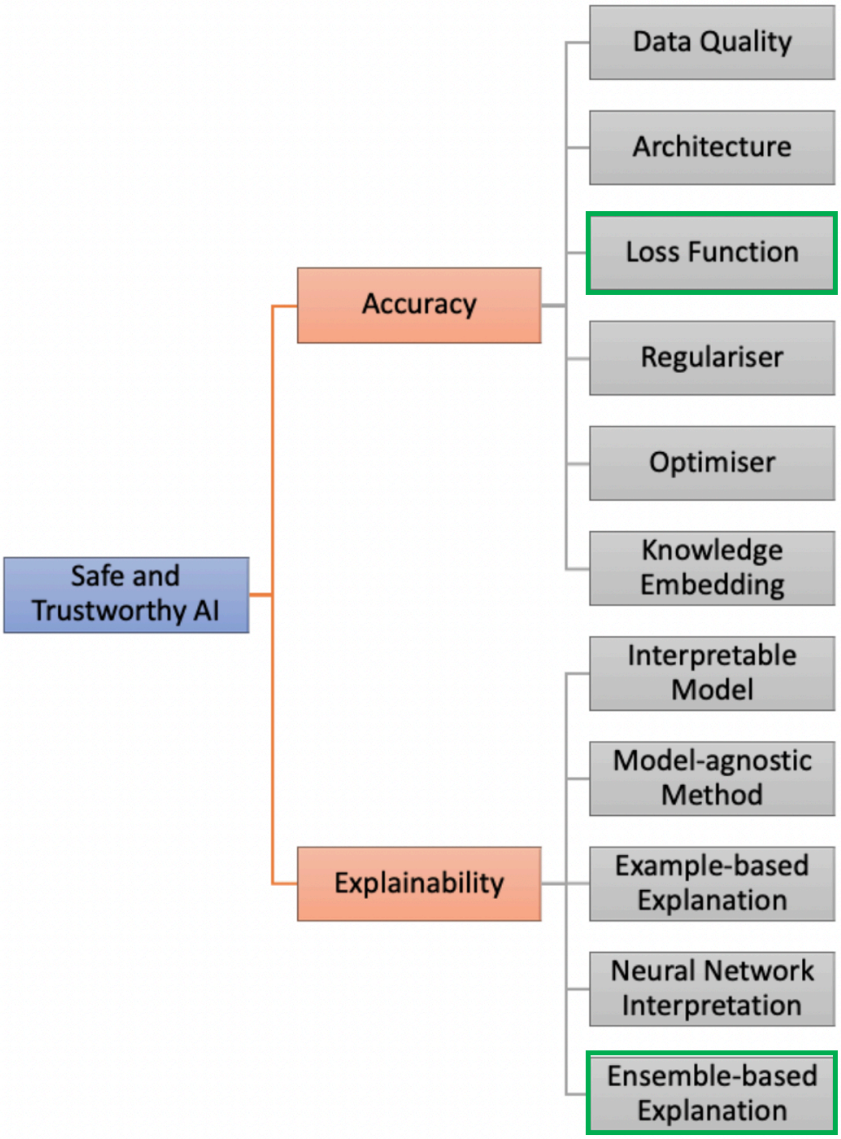



Figure 1.4: Safe and trustworthy Machine Learning requires the predictive model to be accurate and able to explain its output. The prediction accuracy and explainability of Machine Learning models can be improved by working on the different aspects shown in this diagram. The items highlighted with green boxes are the focuses of this thesis.

taken as the objective truth as it might potentially be biased and inaccurate [49] — which is a research opportunity to improve current explainability methods. These paragraphs below outline the current research gaps and shortcomings this thesis aims to address to improve accuracy and explainability of ML methods.

Gaps related to ML prediction accuracy for predictive maintenance:

- Traditionally, the loss functions in DL methods are not domain-specific. For example, a common loss function for regression is the mean-squared error, which gives equal weights to positive and negative errors. For predictive maintenance and, more specifically RUL predictions, the loss function can be augmented to include domain knowledge so that DL methods favour early predictions, thus prioritising safety and supporting the logistics of the supply chain in predictive maintenance.

- Loss functions generally calculate the average errors of prediction across a batch of data or the entire dataset, and subsequently update the parameters of DL accordingly, but they do not focus on instances that are difficult for the models to learn. The DL models can be improved by incorporating the knowledge of difficult data instances in the loss function so the models pay less attention to easy to learn examples and more attention to those instances where the model fails.

Gaps related to explainability of ML for predictive maintenance:

- Although FI calculation approaches assist in interpretation, there is a lack of consensus regarding how features' importance is quantified, which makes the

explanations offered for the outcomes mostly unreliable. A possible solution to address the lack of agreement is to combine the results from multiple FI quantifiers to reduce the variance of estimates and to improve the quality of explanations.

- Estimating the importance of features in ML predictive analytics is often uncertain caused by the usage of different ML models, FI techniques, and subsets of data generate different importance coefficient values, often with different magnitude, for the same features [49]. These uncertainties in identifying the contribution of features needs to be captured for stakeholders to make sound decisions.

- Representation of FI as real number can be misleading and incomprehensible for decision making. For example, a FI value of 0.9 may seem arbitrary and not meaningful. In addition, different FI approaches are likely to assign different importance to features. To have a method of capturing the uncertainties of FI calculations and simultaneously being able to represent it in simple linguistic terms benefits decision making.

## 1.4 Aims and Objectives of Thesis

This thesis aims to develop methods for safer usage of ML in sensor data-driven CBM through increased accuracy in prediction and model explanation. The specific objectives to achieve those aims are discussed below.

Objectives related to improved accuracy:

- To develop and test supervised DL models with new types of explicit knowledge embedded loss functions to improve predictive capabilities for sensor-based data by favouring early prediction in RUL.
- To assess the advantage of implicit knowledge embedded loss functions on RUL of regression and classification-based DL models using gas turbine engine and fault detection in Air Pressure System (APS) respectively.

Objectives related to explainability:

- To develop and test an ensemble FI framework that takes into consideration of multiple FI methods and ML models for decision fusion that is more accurate in FI estimates compared than the existing FI method.
- To develop an understanding of how ensemble FI behaves under varied data conditions. Furthermore, an analysis of model explainability are used to identified important features influencing the RUL and fault in machines.
- Develop a fuzzy-based method to account for the uncertainty within ensemble FI framework to accurately report explain ML models. Additionally, explain the FI in linguistic term for straightforward explanation.
- Compare different approaches of decision fusion in the development of ensemble FI framework.
- Assess the advantages and disadvantages of fuzzy and crisp-based ensemble FI on synthetic dataset and real-world creep prediction dataset.

## 1.5 Contributions

This thesis's main contributions are the development of a regression-based dynamically weighted loss function that improves neural networks' proficiency in predicting RUL. Furthermore, an existing classification-based dynamically weighted loss function — Focal Loss (FL) is investigated for a fault detection in APS task. Additionally, domain knowledge, such as the asymmetries in RUL time predictions are embeded in the FL with the objective to improve its predictive capabilities further. The development dynamically weighted loss function and asymmetric loss function are designed to address the research gap of not utilising non-domain specific loss function for improved DL prediction in predictive maintenance.

For improving explainability in ML models, a new crisp-based ensemble FI framework is created to ensure that the FI quantification is robust and more reliable for safety-critical applications. By combining multiple ML models and FI methods, the framework is designed to address the problem of potentially inaccurate FI quantification resulting from using only on one ML model or FI method. The behaviour of the crisp-based ensemble FI framework is analysed for different data conditions and a case study on a real world data. While the crisp-based ensemble FI framework improved FI quantification accuracy over traditional FI approaches it does not capture the uncertainties produced by the usage of different ML models and FI methods. To overcome the shortcomings of the crisp-based ensemble FI framework, a new framework is built upon the crisp-based ensemble FI framework named Fuzzy Ensemble Feature Importance (FEFI) is developed to capture the uncertainties while explaining the outputs in linguistic term for ensemble FI methods is developed using the

Fuzzy Systems. The core contribution of FEFI is to address the lack of uncertainty estimations in FI quantification and also represents the FI quantification in simple linguistic term. The framework can be applied to both regression and classification-based problems.

## 1.6 Thesis Structure

The thesis is organised as follows:

- In Chapter 2, the concept of loss functions in DL are introduced, followed by the literature review of loss functions in the state-of-the-art DL models. Subsequently, the proposed dynamically weighted loss functions for regression is introduced. The novel loss function focuses on adjusting the loss based on individual instances and it is tested on a gas turbine engine RUL dataset. Additionally, an existing dynamically weighted loss function for classification is also investigated on a air pressure system fault detection dataset. Finally, an asymmetric loss — a special case of the proposed loss function designed to enable early predictions in gas turbine engine RUL is also assessed.

- In Chapter 3, several state-of-the-art FI techniques are reviewed, along with a discussion of their strengths and weaknesses when applied to different data. Subsequently, a novel ensemble FI framework is introduced. The new framework uses multiple FI methods along with multiple predictive models The new ensemble FI method is assessed on synthetic data under varied conditions and real-world case study of identifying features that affects the creep rate of 3D

printed alloys.

- In Chapter 4, a new fuzzy systems-based framework that captures the uncertainties of ensemble FI is introduced. The same framework also enables the output of ensemble FI to be explained in linguistic terms. This new framework is compared to the original ensemble FI method introduced in Chapter 3 on synthetic datasets and a real-world dataset.

- Chapter 5 concludes this thesis. The research work, contributions and findings are summarised. Finally, future work on safer ML for CBM is discussed.

# Chapter 2

# Embedding Knowledge in Loss Functions for Condition-Based Maintenance

## 2.1 Introduction

The increasing number of successful examples of applications of deep learning in manufacturing, automotive, marine, and aerospace industry has shown that it is a viable tool to support data-driven predictive maintenance [50, 51, 52, 53, 54]. Current DL research for these application areas, however, mostly focuses on changing the model architectures to improve RUL or fault prediction accuracy. The literature regarding the improvement of other components of the DL model or training pipeline is scarce. Aspects such as data quality, data augmentation, loss function, and model

regularisation (Figure 2.1) are rarely investigated in CBM research.

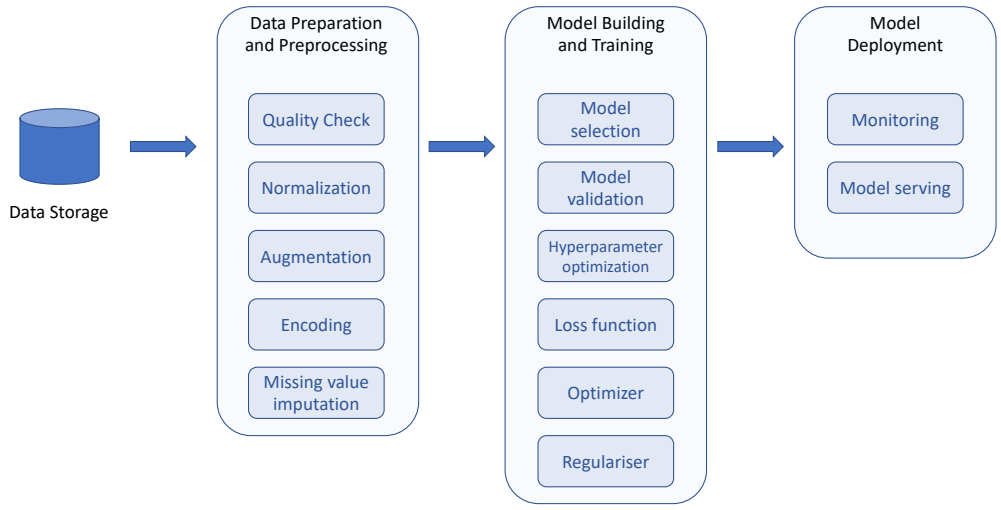



Figure 2.1: Deep learning model pipeline, from data preparation to model training and deployment.

The fundamental building blocks of DL are the neural units, or commonly referred as units. The individual unit has multiple inputs and produces an output using the mathematical combination of those inputs. A neural network consists of three main layers: (1) input (2) hidden and (3) output layer. Each layer consists of one or more neural units. The input layers are the input into the neural network, the hidden layers perform calculations depending on the task, while the output layer produces the final predictions. The different layers are connected using weights and biases parameters. The parameters are iteratively updated through the learning process to produce the desired output with the objective to minimise the prediction error. The learning process uses the error of the predicted output within a feedback loop to refine the parameters until the error is minimised. The calculation of the error

used as a feedback for the network is the loss function. The loss function can be calculated for a single instance, a batch of instances, or even the entire training dataset depending on the design of learning process. A loss function's purpose is to quantify the prediction errors that the model seeks to minimise during training. Commonly used loss functions for regression tasks include Mean Squared Error (MSE), Mean Absolute Error (MAE) and Huber loss [55]; standard loss functions for classification include the Cross-Entropy (CE) [21] and Kullback-Leibler (KL) divergence loss [56].

The loss functions mentioned before are not domain-specific, and they do not implicitly focus on difficult to learn instances. Our hypothesis is that by assigning additional weighting factor to the loss function, it can learn to focus on instances that are difficult to learn, therefore leading to more accurate prediction for RUL and fault detection tasks. An additional hypothesis is that the incorporation of domain knowledge into the loss functions will lead to more accurate RUL predictions. In this thesis, both RUL prediction (Regression) and fault detection (Classification) are considered for weighted loss functions. In addition, the RUL prediction case study is also considered to investigate the effectiveness of embedding domain knowledge into a loss function. The objective of this chapter is to establish the means to improve the standard loss function calculation such as MSE, MAE, and CE to achieve better prediction accuracy by discovering which instances are difficult to learn and focus more on them (implicit) on both RUL and fault prediction tasks. An additional objective is to embed external knowledge such as favouring early prediction in RUL task (explicit) to improve prediction accuracy.

While the standard loss functions are typically adequate for solving prediction

tasks, they can be modified to address problems, such as scarcity of instances in the data space. For example, FL [57] is a modified CE loss that focuses more on imbalanced datasets, whereby misclassification of minority classes are penalised more heavily. Additionally, loss functions can be adapted to solve domain-specific problems, e.g. for improving robot navigation using loss function based on knowledge of spatial image reconstruction [58]. In another study, a multiphase flow simulation is achieved using DL techniques with a custom loss function that imposes local mass conservation constraints to ensure higher physical realism in the simulation [59]. The example of using local mass conservation constraints in the loss function is an example of physical loss function. By adding knowledge of physical law into the loss functions, the model can learn to obey physical laws so as to not make a prediction that is physically impossible. McGowan *et al.* [60] employs similar approach to their porosity prediction in laser metal deposition problems using physics-based custom loss function. Five physics-based loss functions using a combination of temperature, length-to-width ratio of ideal melt pool, and the temperature is used to predict the porosity of laser metal deposition. Results showed that by using the custom loss functions they were able to improve the precision of porosity prediction compared to DL model using the standard categorical CE loss function.

Besides custom physical loss functions, another type of custom loss functions that focus on economic outcome from model prediction is known as enonomic loss function. A notable example of economic loss function is Taguchi loss function [61]. Taguchi loss function evaluates the the quality of a product or its deviation from an ideal product quality. Another perspective of the Taguchi loss function is the

measurement of the deviation of the parameters controlling a process. The process includes but not limited to manufacturing, design of a policy, etc. In a paper by Pandey *et al.* [62], they developed a combined model for joint optimisation for preventative maintenance time interval and quality control parameters using the Taguchi loss function. The Taguchi loss function is used to optimised for (1) sample size for quality check, (2) frequency of quality check, (3) time interval between preventative maintenance check, and (4) control limit. Control limit is the threshold for the condition of an equipment that deviates over the limit safe operability. Overall, they found that by adopting the Taguchi loss for its process leads to better process performance and lower economic loss.

Similarly, custom loss functions can be applied to DL-based predictive maintenance to improve prediction accuracy. Specifically, knowledge embedded loss functions, such as a dynamically weighted loss functions — where the loss function is designed to account for difficult to learn instances by adding a weight to the errors as shown in Figure 2.2. Additionally, asymmetric loss functions — where the loss function guides the DL model based on domain knowledge. Devising new and using existing Dynamically weighted loss function and the asymmetric loss function for predictive maintenance are part of the core contribution of this thesis. The novelty of weighted loss functions is that they automatically learn to focus on difficult-to-learn instances, whereas for asymmetric loss function the novelty is that domain specific knowledge is added to directly improve RUL prediction. Both proposed loss functions are tested on the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) gas turbine engine RUL dataset. An additional case study of fault detec-

tion in APS is also tested on the FL. This chapter is organised as follows: First, the background on the formal learning model and how loss functions work are introduced in Section 2.2. Subsequently, a dynamically weighted loss function is introduced for the regression task in Section 2.3 and validated on CMAPSS RUL dataset in Section 2.6.1. Additionally, the efficacy of FL is tested on APS fault detection dataset in Section 2.6.2. APS is a commonly used system in aerospace application. Finally, an asymmetric loss designed to perform early predictions is introduced in Section 2.4 and validated against the same gas turbine engines RUL dataset in Section 2.6.3. The discussions on the results and conclusions are presented at the end of each case studies.

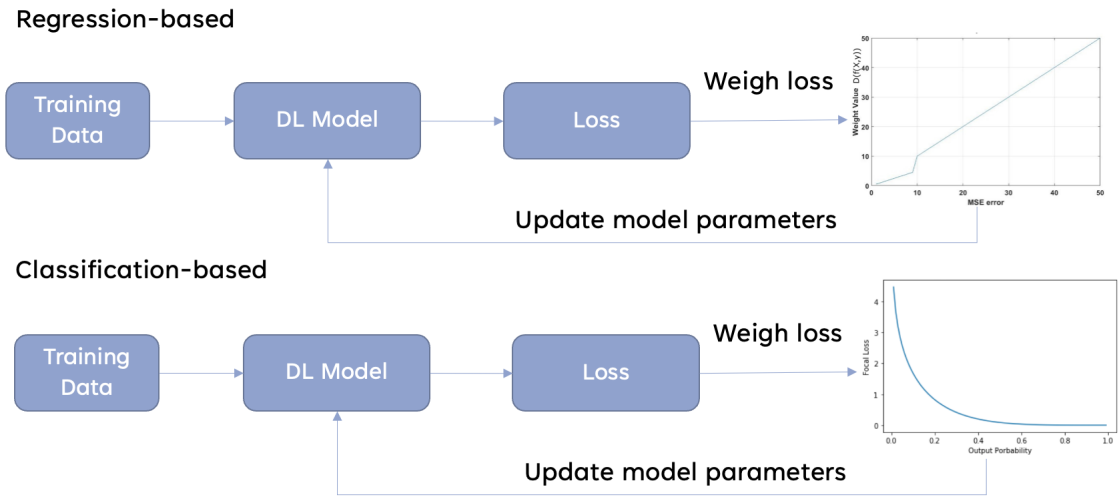



Figure 2.2: An overview of how dynamically weighted loss function works for both regression and classification tasks.

## 2.2 Background

### 2.2.1 Introduction to Formal Learning Model and Loss Function

This section introduces the role of the loss function in a data-centric learning model. Loss functions were initially introduced within the statistical learning theory. Under the statistical learning theory, learning models include the following elements:

1. Feature data, $X$: This is a finite sequence of features, $X = X_1, ...X_n$ that the model tries to label. An individual element of $X$ is known as an instance. The features in the context of CBM are for example, the temperature at the fan inlet in gas turbine engine, the vibration of the turbine blade, etc.

2. Label data, $y$: This is the set of ground-truth values that maps to the feature data instances. Note that this only applies to supervised learning. The labels in the context of CBM are for example, the health indicator, the RUL, fault conditions, etc.

3. Training/Validation/Test data, $S = (X_1, y_1, ..., (X_n, y_n))$: Training data is the pair of feature and label data the model uses to update its parameters. Validation data is used to estimate how well the model has been trained and it is often used to tune the hyperparameters of the model. Furthermore, validation data comes from the same distribution as train data. Test data is unseen data that comes when the trained and tuned model is deployed in production environment and the distribution of the test data is unknown. Generally,

train/validation/test split of the data is done randomly on a data on a 60/20/20 ratio. However, the exact split of data is dependent on each use case.

4. Model's output: A model produces a prediction given new data instances using a function, $h : X \rightarrow y$. This function is called a hypothesis or a predictor. For example, the hypothesis can be a function that inputs the temperature and pressure of an engine and predicts its RUL.

5. Evaluation metrics: The evaluation metrics are functions that calculate the hypothesis's errors. The training error of $h$ is defined as follows:

$$L_{D,f}(h) = \mathbb{P}_{x \sim D}[h(x) \neq f(x)] = D(x : h(x) \neq f(x)) \tag{2.1}$$

Where $L_{D,f}(h)$ is the loss function and the subscript $(D, f)$ indicates that the error is calculated over the probability distribution of x denoted by $D$ and the function that generates the predicted value, $f$. $\mathbb{P}_{x\ D}$ represents the distribution of the error.

Using the training data, $S$, and the loss function, $L(h)$, can be reformulated to the training error, as follows:

$$L_S(h) = \frac{|i \ni [n] : h(x_i \neq y_i)|}{n} \tag{2.2}$$

where [n] = 1,...,n are the data instances. The training error (Equation 2.2) is also known as empirical risk. The learning process's goal is to find a hypothesis, $h$ that minimises $L_s(h)$ and it is a core concept in the formal learning model, known as Empirical Risk Minimisation (ERM). However, due to ERM formulation, it always

finds a hypothesis that minimises the training error. The hypothesis with the lowest possible training error is not always ideal. Low training error could result from overfitting — the hypothesis fits too closely to the training data and is not representative of the true data distribution.

One solution to the overfitting problem is to restrict the search space of ERM learning rule. To achieve this, the learning model chooses a set of finite hypothesis class. Some example of restricting the search space are limiting the precision bits of a system, i.e. the number of decimal points, set a finite number of parameters for a model, and set a finite number of inputs and outputs for a model. Given a sufficiently large training sample (assuming training sample classes are balance) and a finite class of $\mathcal{H}$ then $ERM_{\mathcal{H}}$ does not overfit [63]. Furthermore, using $ERM_{\mathcal{H}}$s the model establishes a hypothesis $h \in \mathcal{H}$ with the lowest possible error. Under the finite hypothesis class, the model is correct up to a level of confidence and error — this is also known as Probably Approximately Correct (PAC) learning. The PAC learning states that $ERM_{\mathcal{H}}$ returns a hypothesis, $h$, such that $L_{(D,f)}(h) \leq \epsilon$ with minimum confidence of $1 - \delta$. $\delta$ is the confidence parameter that indicates the probability of $h$, achieving the minimum required error. In summary, if there are sufficient representative data, the ML model is likely to achieve predictions with minimal error within a certain confidence interval.

The loss function introduced deals solely with the prediction task, $L_D(h : X \rightarrow y)$. To generalise the loss function beyond prediction, a random variable $z$ represents the output of any arbitrary task. The newly generalised loss function is defined as follows:

$$L_D(h) = \mathbb{E}_{z\sim D}[\ell(h, z)] \tag{2.3}$$

Where $\mathbb{E}$ is the expected value of loss as defined by function, $\ell$. The generalised loss function can be adapted for any task at hand. For example, if the task is to predict a continuous number value using squared error, the loss function becomes an expected squared error as follows:

$$L_D(h) = \mathbb{E}_{(x,y)\sim D}(h(x) - y)^2 \tag{2.4}$$

The formal learning model provides a theoretical framework to ensure that learning is possible under the right conditions. Furthermore, the formal learning model shows that the loss function is a foundational component in constructing ML models and it is necessary to understand it before augmenting the loss functions for ML-based CBM. The subsequent section explores how loss function is used in DL models.

### The Role of Loss Function in Neural Network Learning

To illustrate how the learning process of a neural network typically occurs, a simple architecture known as perceptron is employed. A perceptron consists of an input layer with its neural units organised, as shown in Figure 2.3.

Each neural unit is a function that receives a vector of values and outputs a scalar value. The vector input to a neural unit are the features (input data) or inputs $x$ with the network parameters, the weights $w$ and biases $b$, as expressed in Equation (2.5):

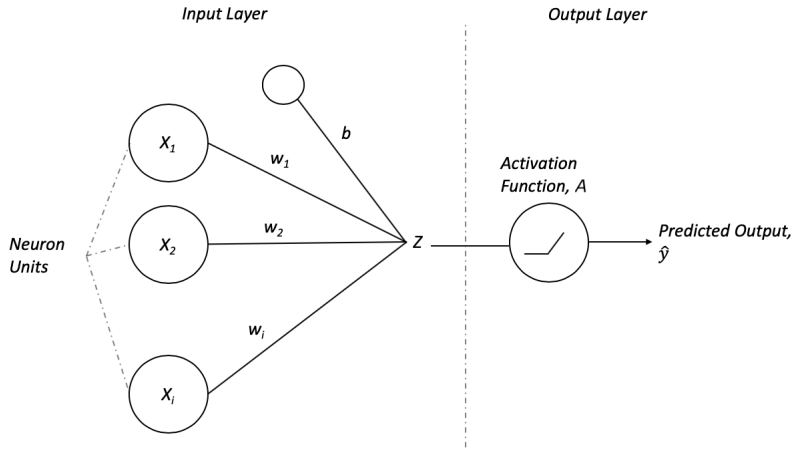



Figure 2.3: Basic components in a perceptron neural network: the input layer takes an arbitrary number of inputs, $s$, the weights, $w_i$ map the inputs to the subsequent layer, a bias, $b$, activation function $A$ to introduce non-linearity into the function and the output, $\hat{y}$.

$$Z_\theta(x_i) = x_i * w_i + b \tag{2.5}$$

where $\theta$ is the weight, $w$ and bias, $b$. The output is the dot product of the inputs and their corresponding weights, and it is followed by adding a constant value bias, $b$. The activation function computes the output. The choice of activation function is highly dependent on the task. The inputs in the context of CBM are the sensors data collected and the output produces a prediction according to the task, such as predicting different failure modes of an aircraft fuel system using input and output fuel pressure of boost pump, fuel flow velocity, and the vanes rotating speed [64].

The perceptron neural network training process starts by a random initialisation of the the weights $w_i$ and bias $b$ variables. A linear activation function is applied, as calculated in Equation (2.6):

$$A_\theta(x_i) = x_i \tag{2.6}$$

Furthermore, the output from activation function is equivalent to the predicted output, $\hat{y_i}$ (Equation 2.7):

$$\hat{y_i} = A_\theta(x_i) \tag{2.7}$$

Once the predicted output, $\hat{y_i}$ is obtained, the error, $E$ of prediction can be evaluated using the perceptron's output against the actual value, $y_i$:

$$E = \sum_{i=1}^{n} [\hat{y_i} - y_i] \tag{2.8}$$

The $E$ calculated in Equation (2.8) is used as an input to calculate the overall

loss. Equation (2.9) is a mean square loss function for this application:

$$L(\theta_i) = \frac{1}{2n} \sum_{i=1}^{n} E^2 \tag{2.9}$$

Subsequently, gradient descent is used to update the weights and biases based on the magnitude of the loss. Gradient descent is an iterative optimisation algorithm used to minimise the loss by updating the weights as shown in Equation (2.10)

$$w_i = w_i - \alpha \frac{\partial}{\partial w_i} L(w_i) \tag{2.10}$$

The partial derivative in Equation (2.10) takes the derivative of the loss function with respect to weight, which is the equivalent to calculating the gradient of the loss. The learning rate $\alpha$ is a parameter that controls the magnitude of change in each iteration. Through the iteration, gradient descent converges on the global minima of the loss curve and provides the best value for each weight parameter as illustrated in Figure 2.4.

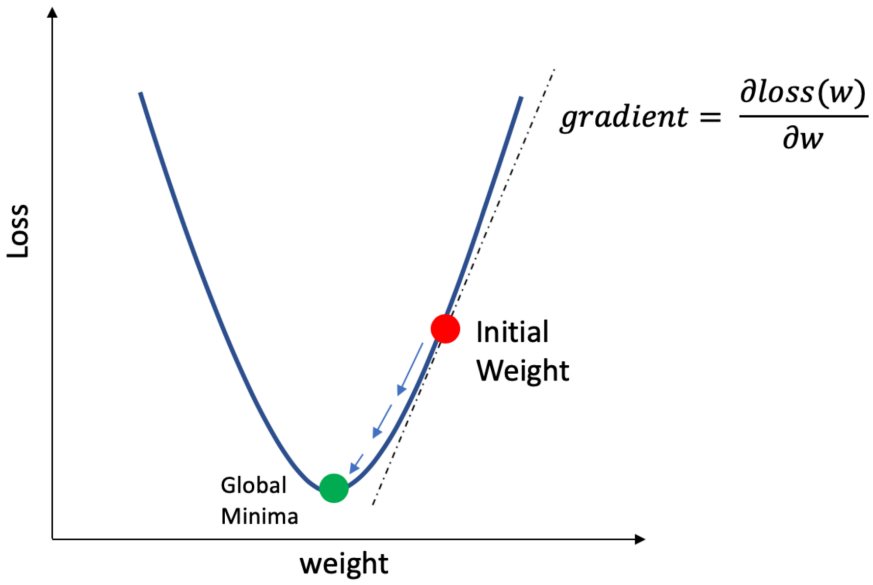



Figure 2.4: An initial weight value iteratively minimised based on the partial derivative of a loss function to achieve global minima in loss.

Figure 2.4 illustrates the importance of the loss function, as it guides the search to achieve the model with the best predictive power. The design of loss functions can have significant outcome on the learning process and ultimately influence the predictive accuracy of the learned model.

### Stochastic Gradient Descent

The Stochastic Gradient Descent (SGD) is a popular optimiser for DL models. SGD is a stochastic approximation of gradient descent discussed in the previous section. It approximates gradient on a single instance rather than an entire dataset. In equations 2.9 and 2.10 from Section 2.2.1, the parameters of a DL model are adjusted iteratively, using a function of sum of all errors. The main idea of SGD is to leverage

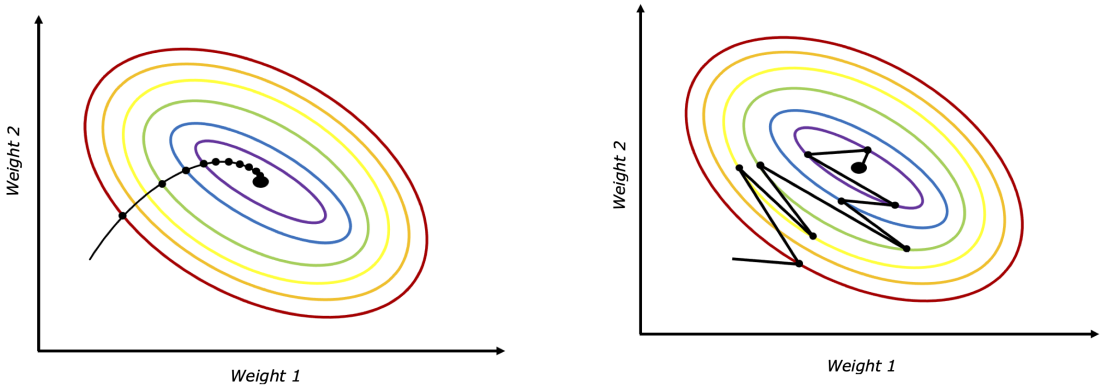
the knowledge that every instance provides useful information that optimises the parameters, therefore, a single random instance every iteration can also minimise the loss given enough iterations. The formula for SGD is defined as follows:

$$w_i = w_i - \alpha L(x_i, y_i) \tag{2.11}$$

where $i$ is a random index from an input instance $i \in [n]$. Intuitively, the SGD follows the descent direction of gradient descent in expectation, and it can converge in fewer iterations as not all instances are used to compute the loss. Figure 2.5 illustrates the descent path of (a) gradient descent and (b) SGD on an MSE loss function. The figure illustrates a top down view of the 3D loss function curve where $weight_1$ and $weight_2$ is the parameters of a model. The axis going into the figure is the loss. The trajectory towards the minimisation of loss (center of each plot) is more haphazard for SGD when compared to gradient descent due to inconsistencies in loss provided by a single instance.

While SGD is likely to converge to a global minima for loss in fewer iterations, there is a theoretical optimality gap between gradient descent and SGD [65]. Optimality gap is the fundamental difference between gradient descent and SGD in its ability to achieve minimum loss. There are several ways to reduce this gap, such as:

- **Learning rate decay**: This is the gradual reduction of the learning rate, $\alpha$ (Section 2.2.1, page 53) after certain iterations. The reduction of the learning rate prevents the gradient to overshoot as the parameters are near the optimal point.




(a) Gradient Descent (b) Stochastic Gradient Descent

Figure 2.5: Trajectory of descent for (a) gradient descent and (b) stochastic gradient descent on a MSE loss function.

- **Minibatching**: Minibatching is a technique to calculate the average of $m$ number of stochastic gradients. This reduces the stochastic gradients' variance, leading to more optimal descent direction with some sacrifice in speed.

- **Momentum**: The idea of momentum in SGD is to use the previous gradient direction to guide the current gradient step. If the previous step's parameters update incurs a significantly lower loss, it is a good idea to continue in the same direction.

The discussions around the formal learning model and loss functions generally focus on convex function, i.e., the squared loss. However, many ML use cases, such as embedding knowledge, might change the loss function from convex into a non-convex function. Non-convex loss function can pose a problem if they contain saddle point and multiple local minima as the gradient can be trapped in these spots. In the next section, how gradient descent can converge on non-convex loss function is discussed.

### 2.2.2 Implicit Convexity

Gradient descent can converge on non-convex loss functions under two conditions [66]: (1) The loss function must satisfy Lipschitz continuity in its first derivative. It means that the non-convex function's gradient can not change too quickly, and it must be differentiable everywhere. (2) All saddle points in the function must be strict saddle. A point is considered a strict saddle if the first derivative is zero ($\nabla f(x) = 0$), the minimum eigenvalue of the second derivative is less than zero ($\lambda_{min}(\nabla^2 f(x)) < 0$), and the maximum eigenvalue is greater than zero ($\lambda_{max}(\nabla^2 f(x)) > 0$). If the two conditions are satisfied, gradient descent is guaranteed to converge to a minimum point.

Similarly, for SGD, there are several conditions necessary to converge under non-convex functions [67]. SGD is non-deterministic as the gradient is now subjected to random sampling from data. To prevent the gradient from moving too rapidly due to randomness, the learning rate, $\alpha$ must be carefully adjusted. Additionally, the condition of the strict saddle is relaxed, only requiring the norm of the first-order derivative of the loss function to be less than or equal to a small value, $\epsilon$, ($||\nabla f \leq \epsilon||$) and the minimum eigenvalue of the second-order derivative to be greater than or equal to the negative square-root of the Lipschitz constant, $\rho$ and, $\epsilon$ ($\nabla^2 f(x) \geq -\sqrt{\rho\epsilon}$). The Lipschitz constant of a function represents the rate of change. If a function has no Lipschitz constant, it is said to be discontinuous. If these conditions are fulfilled, SGD converges. Given the convergence conditions many loss functions that are non-convex can converge and lead to optimal learning models.

### 2.2.3 Commonly Used Loss Functions

Different loss functions are grouped by their tasks, such as regression or classification. Some commonly used loss function and their formulas for regression and classification are shown in Table 2.1.

| | Loss functions | Formula |
|---|---|---|
| **Regression** | Squared loss | $(\hat{y} - y)^2$ |
| | Absolute loss | $\lvert\hat{y} - y)\rvert$ |
| | Huber loss | $\frac{1}{2}(\hat{y} - y)^2, \quad \text{if } \lvert\hat{y} - y\rvert \leq \delta$<br>$\delta\lvert\hat{y} - y\rvert - \frac{1}{2}\delta^2, \quad \text{otherwise}$ |
| | Log-cosh loss | $log(cosh(\hat{y} - y))$ |
| **Classification** | Cross-entropy loss | $-\sum_c^C y_c log(\hat{y}_c)$<br>where C is the number of class |
| | Hinge loss | $max(0, 1 - y * \hat{y})$ |
| | Exponential loss | $e^{-y*\hat{y}}$ |
| | Kullback-Leibler divergence | $\sum_c^C y_c(log(y_c) - log(\hat{y}_c))$ |

Table 2.1: Non-exhaustive list of commonly used loss functions for regression and classification in Machine Learning.

Each loss function exhibits different characteristics, which makes them suitable for different problems. For example, the squared loss has the advantage of fast convergence, and the gradient naturally reduces as loss to zero. However, the squared loss is biased towards outliers as the large errors cause the parameters to change more in response to the large loss. A standard solution to mitigate large loss from biasing

the model is to use an absolute loss or a Huber loss, as they are more robust towards outliers.

While squared loss being biased towards outliers might seem like a disadvantage, it is not so in a domain where classes in data are highly imbalanced such as predictive maintenance where the anomalies are often 'fail' classes. However, none of the loss functions in Table 2.1 are able to implicitly learn which instances difficult-to-learn and they are also not specifically designed for predictive maintenance tasks. In the next section, the dynamically weighted loss function is introduced to improve the standard loss function calculation by focusing on challenging, difficult to learn data instances.

## 2.3 Dynamically Weighted Loss Functions

Dynamically weighted loss functions adjust to difficult to learn instances by adding a higher weight to their losses during training. The error calculated from a dynamically weighted loss function is a mechanism to force the learning process to focus on those instances with the highest error from the deep learning model. The objective is to improve the overall accuracy of the deep learning systems investigated, especially in cases where there is data imbalance. Our hypothesis is that the deep learning models using a dynamically weighted loss function will learn more effectively than a standard loss function in situations commonly found in predictive maintenance, such as data imbalance.

MSE loss function (Equation 2.12) as an example of how to convert a traditional

loss function in a weighted loss function.

$$L(f(x;\theta),y) = (f(x;\theta) - y)^2 \tag{2.12}$$

where $f(x;\theta)$ is the model output, $\theta$ is the model parameter, and x is input while $y$ is the ground truth label or expected output. Next, the MSE is simply multiplied by a weight variable $D$ to be converted to a weighted MSE.

$$L(f(x;\theta),y) = D * (f(x;\theta) - y)^2 \tag{2.13}$$

A large error is an indication of poor learning on a particular instance in the dataset. To place more importance on instances with larger error, the weight variable from Equation (2.13) is updated to a function of $f(x)$ and $y$ as follows:

$$L(f(x;\theta),y) = D(f(x;\theta),y) * (f(x;\theta) - y)^2 \tag{2.14}$$

The specific weight used here scales according to the following condition,

$$D(f(x;\theta),y) = \begin{cases} \frac{|f(x;\theta)-y|}{2} & \text{if } |f(x;\theta) - y| \text{ is } < \text{C} \\ |f(x;\theta) - y| & \text{otherwise} \end{cases} \tag{2.15}$$

The weighting is halved when the absolute difference between predicted value and ground truth is less than a particular constant, $C$. In addition, the weight function

can be generalised to take in different inputs on different loss functions.

$$L(\theta, f(x;\theta), y)' = D(\theta) * L(f(x;\theta), y) \tag{2.16}$$

The flow diagram of the data from input to the new loss function is shown in Figure 2.6.

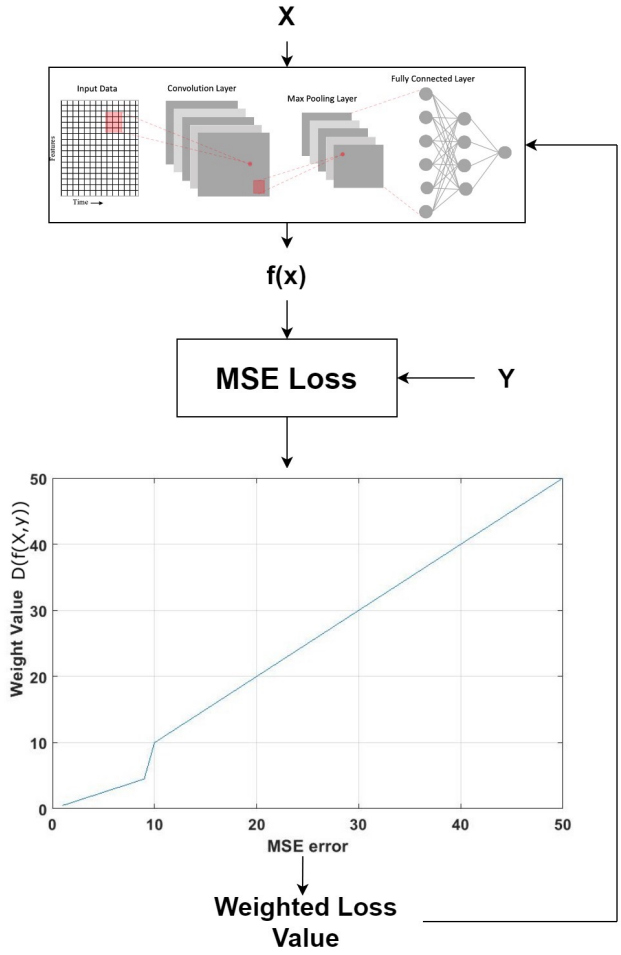



Figure 2.6: The output $f(x)$ from a deep learning model and the ground truth $Y$ are used to calculate the mean square error (MSE) for one instance. The MSE is then passed through a non-linear function to produce the weight that will be used to dynamically adjust the loss function.

It is important to note that one downside of the dynamically weighted loss function is that when used in conjunction with mini-batch stochastic gradient descent or full batch gradient descent. For example, in a batch size of 10, only 1 instance is predicted poorly by the model while the remaining 9 was predicted accurately the increase in loss due to dynamic weighting affects the 9 accurately predicted sam-

ple as well. This could lead to instability in performance and smaller batch size is encouraged during training.

The next section explores the explicit embedding of domain knowledge to further improve the dynamically weighted loss function.

## 2.4 Asymmetric Loss Functions

In the previous section, by using dynamically weighted loss functions the knowledge of instances that requires more attention by the DL models is calculated implicitly. In this section, a way to explicitly embed domain knowledge into the loss function that enable smarter and deterministic bias in DL models that benefits RUL prediction by leveraging the knowledge that early predictions are more favourable compared to late predictions is introduced. The weighted loss function described in Section 2.3 is extended to asymmetric loss, to investigate how predictions of engine RUL in aerospace are affected. Within prognostics and health management, the main metric used to evaluate DL RUL predictions as mentioned in the Section 2.6.1 is the scoring function. The hypothesis is that by using asymmetric loss functions the DL models will be biased towards early prediction and therefore optimises for lower score. In order to investigate the hypothesis, 4 different asymmetric loss functions for regression, i.e, Mean Square Logarithmic Error-Mean Square Error, Linear-Mean Square Error, Linear-Linear, and Quadratic-Quadratic are tested and evaluated on how much they affect different DL architectures performance.

### 2.4.1 Asymmetric Loss Functions Calculations

For regression tasks the typical loss functions are the Absolute Error (AE) loss and the squared error loss. Given x as an array of attributes (or independent variables) within the training set, y as the dependent variable and $\hat{y}$ the deep neural network predictions for y, the absolute error function $f_{AE}$ is calculated as the difference between actual and predicted values, as follows:

$$f_{AE}(\hat{y}_i, y_i) = |\hat{y} - y| \tag{2.17}$$

The square error loss function $f_{SE}$ is determined by the following equation:

$$f_{SE}(\hat{y}_i, y_i) = (\hat{y} - y)^2 \tag{2.18}$$

Using $f_{AE}$ as an example, it can transform the loss function from (2.17) to a weighted loss function, by multiplying it with a weight $a$, as follows:

$$f_{weightedAE}(a, \hat{y}_i, y_i) = a|\hat{y} - y| \tag{2.19}$$

The weight, $a$ enable the control of loss's magnitude by changing gradient of loss function. Additionally, the weighted $f_{AE}$ is redefined as follows:

$$f_{weightedAE}(a, \hat{y}_i, y_i) = \begin{cases} -a(\hat{y} - y) & \text{if } d \text{ is} \leq 0 \\ a(\hat{y} - y) & \text{otherwise} \end{cases} \tag{2.20}$$

Where $d = \hat{y}_i - y_i$. To modify the weighted loss function from (2.20) to an asymmetric

loss function, another weight parameter, $b$ is assigned when $\hat{y}_i - y_i > 0$ as follows:

$$f_{AsymmetricLoss}(a, b, \hat{y}_i, y_i) = \begin{cases} -a(\hat{y} - y) & \text{if } d \text{ is} \leq 0 \\ b(\hat{y} - y) & \text{otherwise} \end{cases} \tag{2.21}$$

In cases when $a \neq b$ the function represented in (2.21) becomes a LIN-LIN asymmetric loss function.

The DL models are then implemented using the asymmetric loss function instead of the traditional symmetric loss functions, such as MSE and MAE. In addition to the LIN-LIN loss functions, 3 other asymmetric loss functions are also employed in case studies discussed in the later part of this chapter, namely, LIN-MSE, MLSE-MSE, and QUAD-QUAD, defined as follows:

$$LIN\text{-}MSE = \begin{cases} -a(\hat{y} - y) & \text{if } d \text{ is} \leq 0 \\ \frac{\sum_{i=1}^{N}(\hat{y_i} - y_i)^2}{N} & \text{otherwise} \end{cases} \tag{2.22}$$

$$MSLE\text{-}MSE = \begin{cases} \frac{\sum_{i=1}^{N}(\log \hat{y_i} - \log y_i)^2}{N} & \text{if } d \text{ is} \leq 0 \\ \frac{\sum_{i=1}^{N}(\hat{y_i} - y_i)^2}{N} & \text{otherwise} \end{cases} \tag{2.23}$$

$$QUAD\text{-}QUAD = \begin{cases} 2a(\hat{y}_i - y_i)^2 & \text{if } d \text{ is} \leq 0 \\ 2(a + (1 - (2a)))(\hat{y}_i - y_i)^2 & \text{otherwise} \end{cases} \tag{2.24}$$

The behaviour of the different loss functions investigated is illustrated in Fig 2.7. From Fig 2.7, it can be seen that when the $error < 0$ ($error = \hat{y} - y$), the loss values are lower compared to $error \geq 0$ using asymmetric loss functions. The asymmetric loss functions therefore allow more severe penalisation for late RUL prediction.

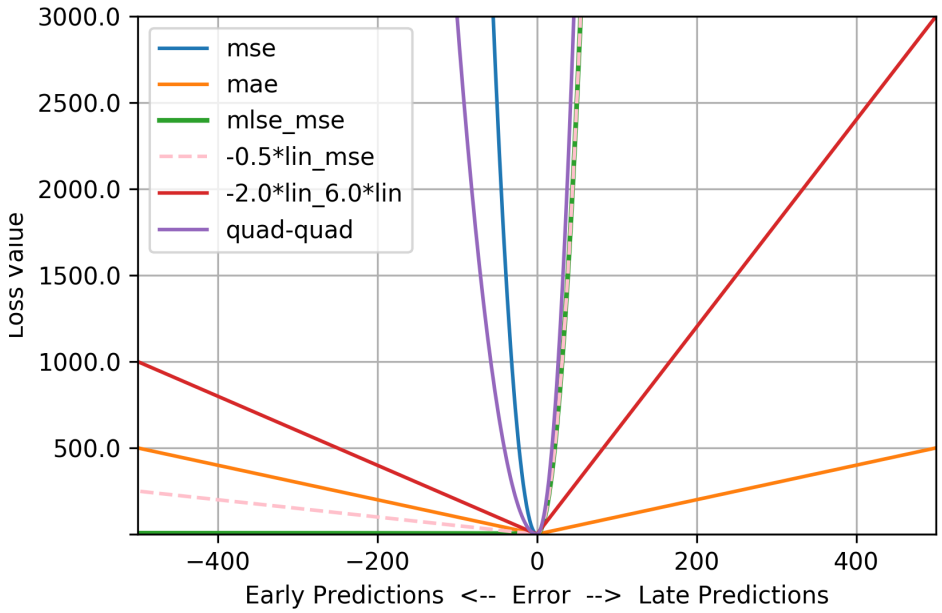



Figure 2.7: Symmetric loss functions such as MSE and MAE compared to different asymmetric loss functions, namely MLSE-MSE, LIN-MSE, and LIN-LIN. The numerical value -0.5, -2.0, and 6.0 represents the weights, $a$ and $b$ of the linear part of LIN-MSE and LIN-LIN loss.

MLSE-MSE is a loss function that limits its penalisation if the error is in negative. In the context of predictive maintenance, the MLSE-MSE only adjust the weights of the model in the same manner as MSE when it is predicting late RUL. When it is predicting early RUL however it only adjust weights of the model if the prediction is extremely early. MLSE-MSE asymmetric loss function is suitable if the tasks is

to always predict early RUL rather than actual RUL. LIN-MSE and the LIN-LIN loss function of the different $a$ and $b$ values focuses on reducing the loss when the model predicts an early RUL compared to late RUL predictions. If the prediction of RUL is too early the loss is still large enough to adjust model to predict RUL more accurately. In the next section, the application of the dynamically weighted loss function, FL, and asymmetric loss functions are investigated in their respective case studies.

## 2.5 Methodology

In this section, the methodology to assess the effectiveness of both dynamically weighted and asymmetric loss functions are introduced. Two case studies is presented for dynamically weighted loss functions. The first is using the proposed dynamically weighted loss function on a gas turbine engine RUL and the second case study is using FL function on an APS dataset. The first step is to explore the data investigated and then perform data preprocessing so the data is in a suitable format for training. Subsequently, each DL architectures as shown in Section 2.5.1 are trained using optimised hyperparameters. The performance of the trained models is evaluated against its respective metrics. Finally, the results are presented and discussed.

### 2.5.1 Deep Learning Architectures Investigated

#### Deep Feedforward Neural Network

Early form of feedforward neural network are Multi-Layer Perceptron (MLP). A MLP consists of three layer types, namely the input, hidden, and the output layer. Each layer is composed of several neural units. The neurals in each layer are fully connected to the neurals in the subsequent layer and the connection holds a weight value that will contribute the output value. The connections' weights are randomly initialised and then updated using the gradient descent method during training. As shown in Figure 2.8, the MLP neural network can be extended to a deep neural network by increasing the number of hidden layers, which allows for learning more complicated relationships between inputs.

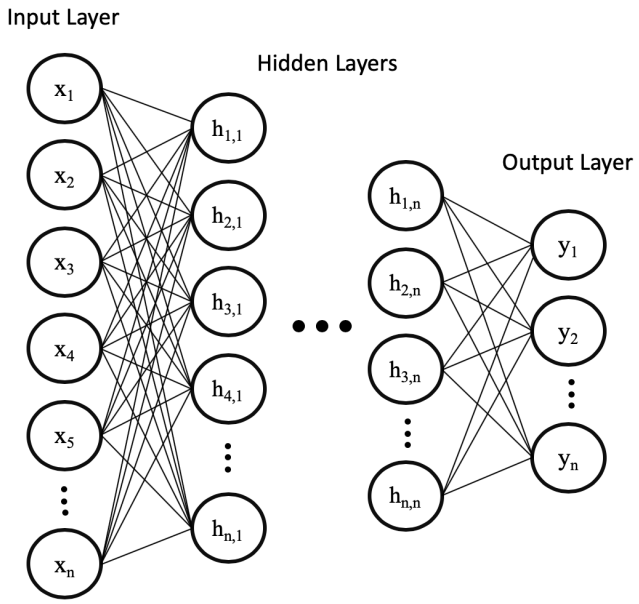



Figure 2.8: A deep feedforward neural network (DNN) similar to the perceptron has the input layer along with the output layer. However, the DNN has a large number of hidden layers and neural units.

### Convolutional Neural Networks

Convolutional neural networks (CNN) [68] are neural networks that contain different layers such as the convolution, max pooling, and fully connected layer. The purpose of max pooling layer is to downsample the input and reduce the dimension while the convolution layer extracts high-level features from the input. This allows CNN to perform better on data that has high spatial correlation with its neighbourhood data-points. Figure 2.9 shows how the spatial relationship within data are preserved through convolution and max pooling using a filter. Furthermore, CNN1D uses a filter that is the same height as the input and the convolution operation occur in a single direction as illustrated in the bottom part of Figure 2.9.

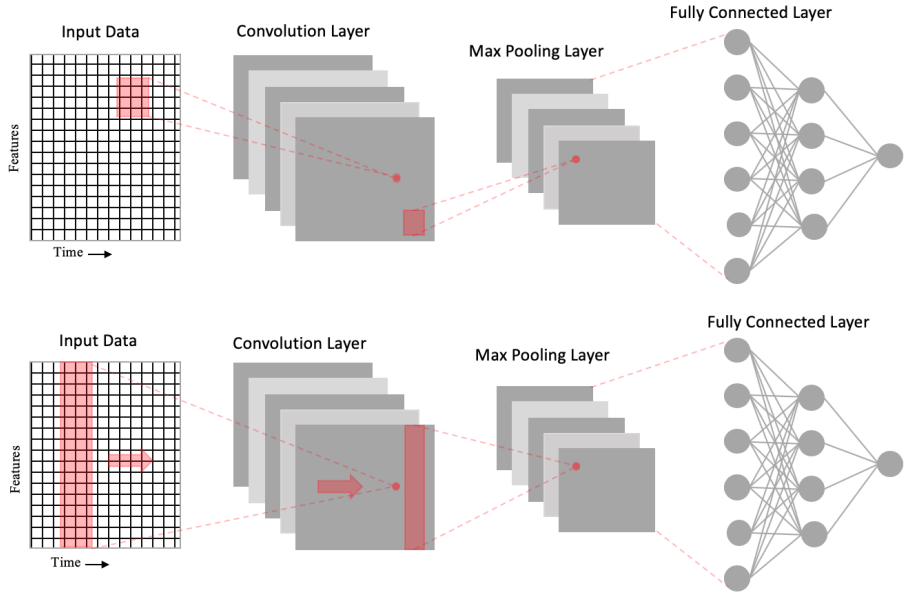



Figure 2.9: A schematic of a 1D and 2D Convolutional Neural Network The **top** part of the figure illustrates a normal two dimensional convolutional neural network (CNN) with its convolutional layer and max pooling layer. The max pooling layer is subsequently flattened to feed the data into a fully connected layer. The **bottom** figure is a one-dimensional convolutional neural network (CNN1D) network where the filter is moving in only one direction to perform the convolution and max-pooling operations.

### Long Short-Term Memory

The Long Short-Term Memory (LSTM) network [69] is a variant of the Recurrent Neural Network (RNN) [70] designed with chain units consisting of input, forget, and output gates as shown in Figure 2.10. Gates are responsible for regulating what information is passed through to the next unit. The input gate controls the influence

of the current input. The forget gate within each unit controls how much information needs to be retained. The output gate controls whether the flow is passed on to the next LSTM unit. This architecture allows for the learning of data with long-term dependencies. Furthermore, a bidirectional LSTM as shown in Figure 2.11 connects two hidden layers of LSTM in the opposite direction to increase the information available to the network by using the past and future states.

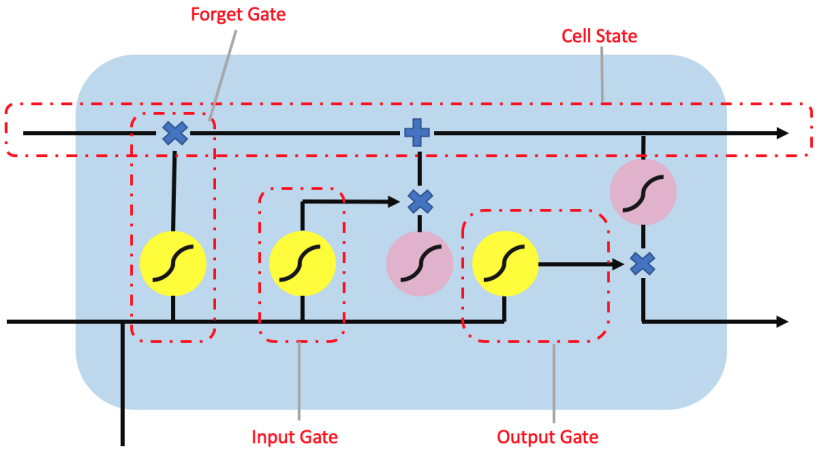



Figure 2.10: The long short-term memory (LSTM) unit contain a forget gate, output gate and input gate. The yellow circle represents the sigmoid activation function while the pink circle represents a tanh activation function. Additionally, the "x" and "+" symbols are the element-wise multiplication and addition operator.

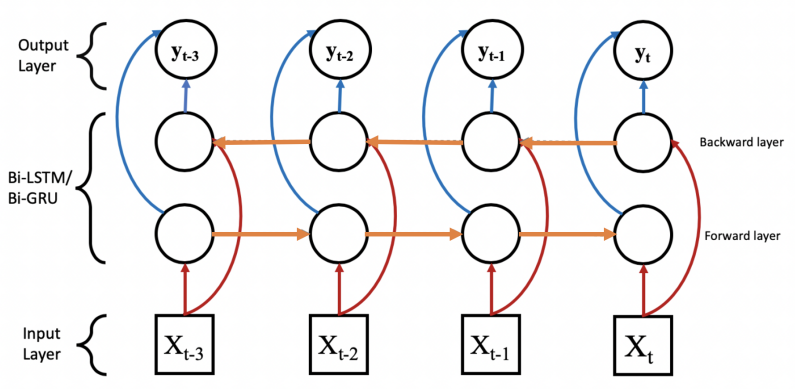



Figure 2.11: The Bi-LSTM and Bi-GRU are structurally the same except for the LSTM and GRU unit. The red arrows indicate the input value flow, blue arrows are the output values, and the grey arrows represent the information flow between the LSTM/GRU units.

### Gated Recurrent Unit

The Gated Recurrent Unit (GRU) [71] is proposed as alternative to LSTM. While GRU and LSTM are similar, they differ in the number of parameters and type of gates. GRU uses only two gates, as shown in Figure 2.12. The two gates are (1) the reset gate to control the memory retention from previous unit and addition of new memory into the unit and (2) the update gate to control input and to remove new information. Therefore, GRU has fewer parameters in its design than LSTM, thus reducing the model complexity while improving on computational efficiency. Similar to LSTM, GRU can also be extended to a Bi-directional GRU architecture.

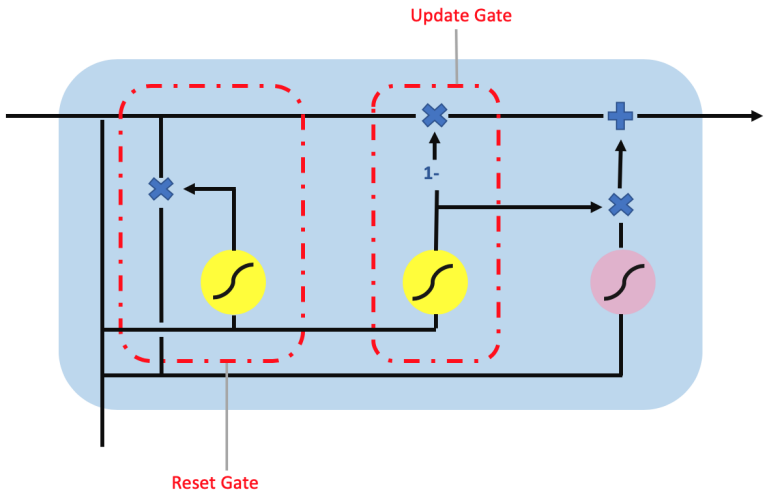



Figure 2.12: The gated recurrent unit (GRU) unit contain a reset gate and update gate. The yellow circle represents the sigmoid activation function while the pink circle represents a tanh activation function. Additionally, the "x", "+", and "1−" symbols are the element-wise multiplication, addition, and inversion operator.

### 2.5.2 Summary of Data Preprocessing

In Table 2.2 a summary of the data preprocessing employed is listed.

| **Task** | **Description** |
|---|---|
| Data split ratio | 80/10/10 (train/validation/test) |
| Data split method | Randomly split between train/validation/test |
| Scaling method | All features scaled between 0-1 |
| Feature Selection | None |
| Filling missing values | Mean |
| Anomaly data | Not removed (anomalies are usually data of interest.) |
| Data encoding | One-hot |

Table 2.2: A table of data preprocessing methods for CMAPSS and APS case studies for dynamically weighted and asymmetric loss functions

## 2.6 Case Studies

### 2.6.1 Case Study on Dynamically Weighted Loss Function: Remaining Useful Life Prediction of Gas Turbine Engine

The first case study focuses on using regression-based dynamically weighted loss function to predict the RUL of gas turbine engine degradation. The goal is to compare the predictive accuracy of DL models with and without dynamically weighted loss. The base loss function employed in this case study is MSE. Furthermore, the weighted and non-weighted loss functions are used in four different DL architectures described in Section 2.5.1. Finally the RUL prediction performance of the DL models using a MSE loss function and dynamically weighted MSE is compared. The hypothesis is that by incorporating the dynamic weighting into the MSE the DL models are able to adjust its parameters accordingly for difficult-to-learn instances leading to higher predictive accuracy compared not loss functions that are not dynamically weighted.

#### Data Generation

The gas turbine engine degradation data used in this case study is CMAPSS by Saxena and Goebel [72]. The data is obtained from a high fidelity simulation of a complex thermo-dynamical system that closely models a real aerospace engine. The failure of the simulated engine is initiated through random point of deterioration. The deterioration continues with increasingly worsening effect. This is modelled after the Arrhenius Model, Coffin-Mason Mechanical Crack Growth Model, and Eyring

Model [72, 73]. The commonality between the three deterioration models listed are the exponential behaviour of fault evolution. The system measures the loss of efficiency and flow until a failure criterion is reached. The degradation and damage propagation trends are modelled as follows:

$$h(t) = 1 - exp(a(t)t^{b(t)}) \tag{2.25}$$

Equation (2.25) is a generalised Equation for the health index of gas turbine engine, $h(t)$ with respect to time. Subsequently, the system includes a non-zero initial degradation, $d$. The initial non-zero degradation is common in real-world systems due to manufacturing inefficiencies or error:

$$h(t) = 1 - d - exp(a(t)t^{b(t)}) \tag{2.26}$$

The decay of efficiency, $e(t)$ and the loss of flow, $f(t)$ are described using (2.26), as follows:

$$e(t) = 1 - d_e - exp(a_e(t)t^{b_e(t)}) \tag{2.27}$$

$$f(t) = 1 - d_f - exp(a_f(t)t^{b_f(t)}) \tag{2.28}$$

Efficiency and flow are modelled separately as different faults exhibit different trajectories of degradation for each of the terms. The terms from (2.27) and (2.28) are combined to form the final model of damage propagation in gas turbine engine:

$$H(t) = g(e(t), f(t)) \tag{2.29}$$

Using the damage propagation model in Equation (2.29) the data is generated as follows:

1. Define initial deterioration parameters, $e_0$ and $f_0$ for (2.27) and (2.28).

2. Impose an exponential rate of change for flow and efficiency loss for each data set, denoting an otherwise unspecified fault location with increasingly worsening effect by setting the parameter $a$ and $b$ in (2.26).

3. Stop when failure criterion (loss of flow and efficiency) is reached. The failure criterion in this case is when $H(t) = 0$.

4. Add mixture noise model into final output data reflect real world scenario. Additionally, the feature data are collected from the sensors measurements listed in Table 2.3. By combining the output and feature data, the complete dataset can be obtained for training and testing DL models to perform RUL prediction.

### Data Description

The CMAPSS dataset contains training and test sets with 6 different operating conditions. The training set comprises of the complete engine life cycle data (run until failure), while the test set has a RUL range of 10 to 150 cycles. The training data consist of 100 engines with a total of 20631 cycle, while the test data consist of

Table 2.3: Description of the CMAPSS dataset sensor features.

| Symbol | Description | Unit |
|---|---|---|
| T2 | Total temperature at fan inlet | °R |
| T24 | Total temperature at Low Pressure Compressor outlet | °R |
| T30 | Total temperature at High Pressure Compressor (HPC) outlet | °R |
| T50 | Total temperature at Low Pressure Turbine outlet | °R |
| P2 | Pressure at fan inlet | psia |
| P15 | Total pressure in bypass-duct | psia |
| P30 | Total pressure at HPC outlet | psia |
| Nf | Physical fan speed | rpm |
| Nc | Physical core speed | rpm |
| epr | Engine pressure ratio (P50/P2) | — |
| Ps30 | Static pressure at HPC | psia |
| phi | Ratio of fuel flow to Ps30 | pps/psi |
| NRf | Corrected fan speed | rpm |
| NRc | Corrected core speed | rpm |
| BPR | Bypass Ratio | — |
| farB | Burner fuel-air ratio | — |
| htBleed | Bleed Enthalpy | — |
| Nf_dmd | Demanded fan speed | rpm |
| PCNfR_dmd | Demanded corrected fan speed | rpm |
| W31 | High Pressure Turbine coolant bleed | lbm/s |
| W32 | Low Pressure Turbine coolant bleed | lbm/s |

100 engines with a total of 13096 engine cycles. The datasets contain the engine unit number, the operating cycle number of each unit, the operating settings and the raw sensor measurements.

### Data Preprocessing

CMAPSS consists of 3 operation settings and 21 sensors features (Table 2.3). However, a total of 7 features are discarded as they remained constant throughout the gas

turbine engine degradation process and provided no useful information. Therefore only the sensors with the following indices are select: 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, and 21. Additionally, the data is normalised between $[0, 1]$ to ensure that each feature is represented equally in the learning process. Subsequently, the labels are preprocessed. The labels are the remaining RUL cycle for each instance of the data and each complete cycle is degraded linearly. Since the fault does not occur at the early stages of engine cycle, the value of the maximum cycle is capped at 100 and remains constant until degradation occurs, as shown in Figure 2.13. This allows for the models to differentiate between the healthy state, a RUL of 100 and under degradation, and a RUL cycle smaller than 100.

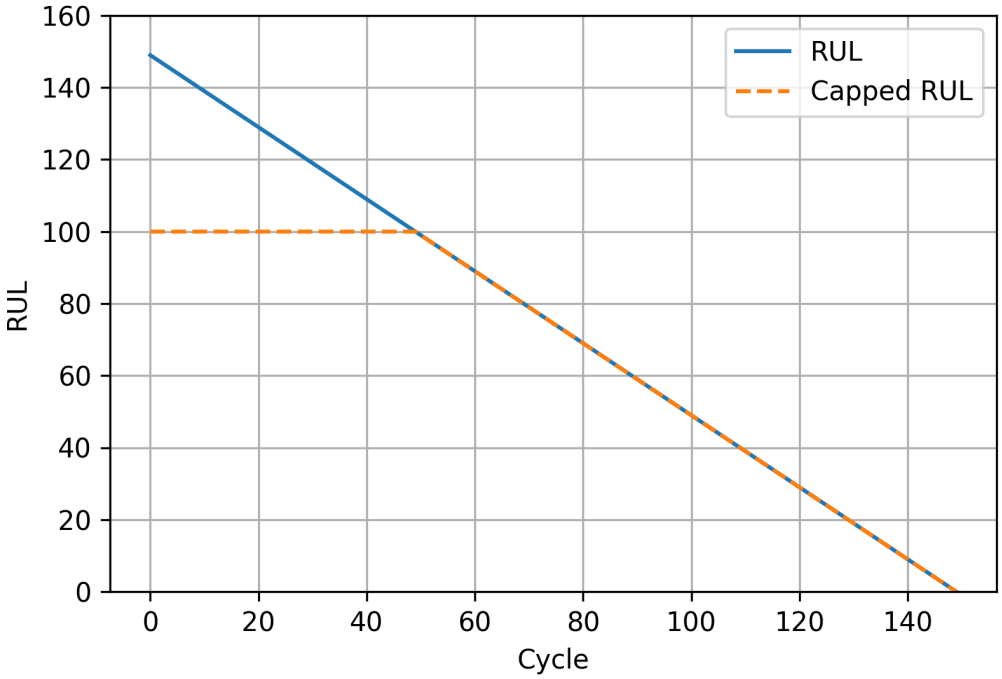



Figure 2.13: Maximum RUL of gas turbine engine are capped to 100 cycles to distinguish the between the healthy state (RUL of ¿=100 cycles) and degradation (RUL of ¡100 cycles) state during preprocessing.

### Deep Learning Architectures Investigated

The following DL model architectures are employed to test the dynamically weighted MSE loss function from Equation (2.16) (page 64), (1) bidirectional LSTM, (2) DNN, (3) CNN1D, and (4) bidirectional GRU as explained in Section 2.5.1 (page 71). The hyperparameters of each DL model employed are listed in Table 2.4. The hyperparameters are optimised using random search.

A L2 regulariser is added to the layers of all models shown in Table 2.4 to reduce overfitting. In the context of neural network, an L2 regulariser is mathematically

Table 2.4: Hyperparameters of all models used to test the new loss function presented in Section 2.3.

| **Deep Learning Architecture** | **Hyperparameters** |
|---|---|
| Bi-LSTM | Number of layers: 2 |
| | Layer 1 units: 100 |
| | Layer 2 units: 50 |
| | Activation function: Leaky ReLU |
| DNN | Number of layers: 6 |
| | Layer 1 units: 100 |
| | Layer 2 units: 500 |
| | Layer 3 units: 100 |
| | Layer 4 units: 250 |
| | Layer 5 units: 12 |
| | Layer 6 units: 6 |
| | Activation function: ReLU |
| CNN1D | Number of layers: 2 |
| | Layer 1 units: 64 |
| | Layer 2 units: 64 |
| | Activation function: ReLU |
| | Filter size: 3 x Features |
| Bi-GRU | Number of layers: 2 |
| | Layer 1 units: 100 |
| | Layer 2 units: 50 |
| | Activation function: Leaky ReLU |

equivalent to weight decays. The L2 regulariser prevents overfitting by limiting the complexity of the network through the penalisation of larger weights, which keeps the weights smaller. Additionally, a dropout [44] rate of 0.5 is also added to all models tested to mitigate overfitting. Dropout is a technique for regularising the network by randomly setting the output to zero (equivalent to setting the weight of the unit to 0).

### Evaluation Metrics

Performance evaluation is the key step to identify and compare the performance between different methods. NASA published a preferred method of performance evaluation for CMAPSS using the idea of asymmetric scoring [72]. In the context of predictive maintenance, it is desirable to predict the time of failure early. Therefore, the scoring is asymmetric around the true time of failure, such that late predictions are more heavily penalised than early predictions. Early prediction can be thought of as a false positive, while late prediction is a false negative. The asymmetric scoring function is defined as follows:

$$\text{Scoring Function} = \begin{cases} \sum_{i=1}^{n} e^{\frac{-d}{10}} - 1 & \text{if } d < 0 \\ \sum_{i=1}^{n} e^{\frac{d}{13}} - 1 & \text{if } d \geq 0 \end{cases} \tag{2.30}$$

where $d$ is $f(x) - y$. Figure 2.14 shows the asymmetricity of the scoring function from Equation (2.30), in which higher penalty scores are assigned to late predictions.

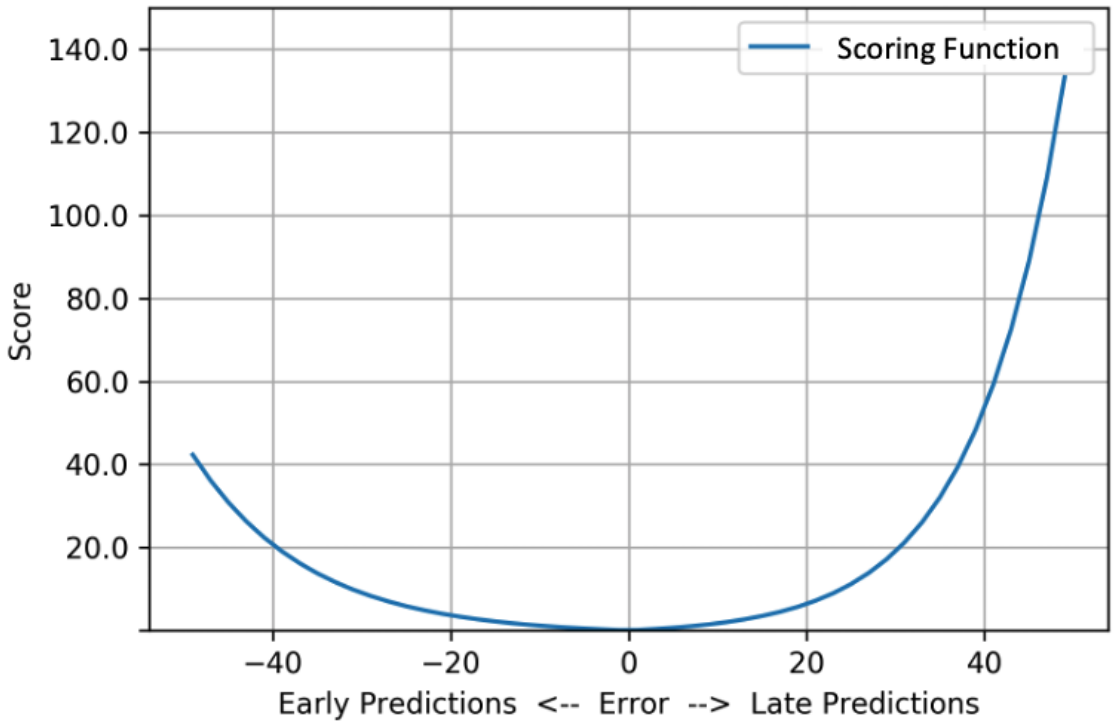



Figure 2.14: Score value as the error increases. The score is calculated using the scoring function (Equation (2.30)), where the early predictions (negative errors) receive lower penalisation.

In addition, Root Mean Squared Error (RMSE) is employed as the second evaluation metrics:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2} \tag{2.31}$$

The scoring function (SF) along with RMSE provide a suitable measure of the different DL models' accuracy. The evaluation metrics alone do not indicate if a result improvement is statistically significant or not. Therefore, the Mann–Whitney–Wilcoxon non-parametric test is used at a 0.05 significance level to test if result improvements

are significant. The Mann–Whitney–Wilcoxon non-parametric test is chosen as the two results distributions are independent. The experiment is replicated 20 times to ensure that the results are more reliable and accurately represent the true distribution of the results.

### Results and Discussion

Table 2.5 shows the comparison between the RUL prediction results of DL models with and without dynamically weighted loss function. Dynamically weighted loss function improved the scoring function's values for all models tested. Using RMSE, Bi-LSTM and CNN1D showed improved performance, while DNN and Bi-GRU's result were inferior. The DNN and Bi-GRU models with dynamically weighted loss function predicted earlier RUL, which caused the predicted output to differ from the ground truth; however, an improvement is still observed in scoring function. This is due to the scoring function's asymmetric property that resulted in the score favouring an early RUL prediction. The results shown in Table 2.5 are the median values, and they do not include outliers.

Table 2.5: Scoring function and root mean squared error (RMSE) before and after using DW for loss function while maintaining the architecture of DL models. Blue coloured text indicates improved performance while red coloured text indicates worsened performance. The values in this table are the median values across 20 experimental runs.

| **Deep Learning Architecture** | **Scoring Function** | **RMSE** |
|---|---|---|
| Bidirectional LSTM | 178.568 | 20.1 |
| Bidirectional LSTM + DW | 129.089 | 13.9 |
| | −27.7% | −30.6% |
| DNN | 93,473.3 | 23.1 |
| DNN + DW | 13,741.3 | 23.9 |
| | −85.2% | +3.4% |
| CNN1D | 112.858 | 22.3 |
| CNN1D + DW | 63.002 | 21.1 |
| | −44.1% | −5.7% |
| Bidirectional GRU | 169.550 | 11.6 |
| Bidirectional GRU + DW | 81.899 | 12.9 |
| | −51.6% | +11.8% |

Figure 2.15 shows that the improvement made by using the new loss function is statistically significant. The number of "*" in Figure 2.15 represents the $p$-value where "***" $< 0.001$, "**" $< 0.01$, "*" $< 0.05$. Figure 2.15 shows that the four models, DNN, Bi-GRU, CNN1D and Bi-LSTM results improvement are statistically significant. Bi-GRU has a $p$-value of $< 0.001$, DNN and CNN1D has a $p$-value of $< 0.01$, while Bi-LSTM has a $p$-value of $< 0.05$. In addition, anomalies are observed in the results shown in the boxplots. This is caused by the random initialisation of the initial weights, which resulted in the variability of the final output. Therefore it is important to run the experiment multiple times to ensure that the actual distribution of the final output is captured.

### 2.6.2 Case Study on Dynamically Weighted Loss Function: Fault Detection in Air Pressure System

The second case study investigates the performance of dynamically weighted loss functions in the context of classification. The classification task in this case study is the fault detection of APS. The same DL model architectures from the previous case study are employed here, but the choice of loss function is different. Cross-entropy, FL, and the dynamically weighted version of cross-entropy are investigated.

#### Data Description

The function of an APS is to produce pressurised air for braking and gear changes. Therefore, it is important that the APS' fault are accurately detected, as a miss in fault could lead to undesirable outcomes. The APS failure data has a total of

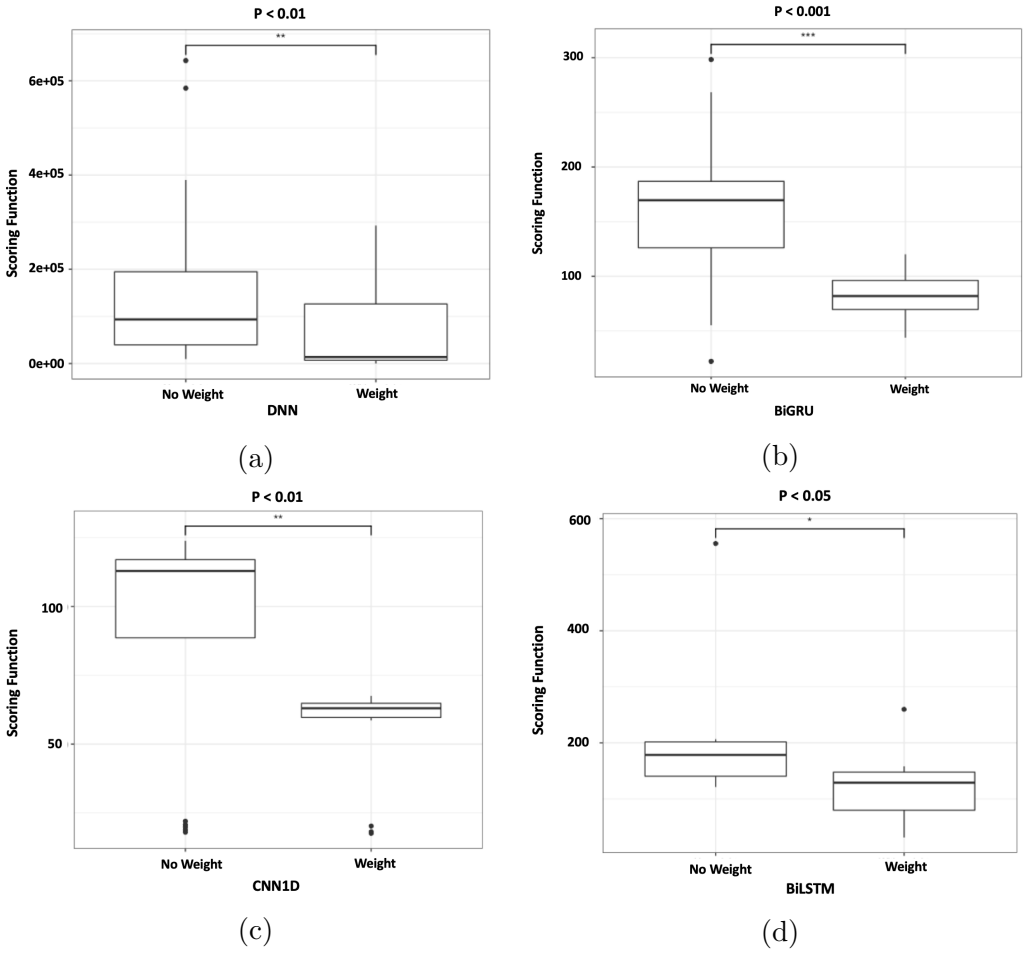



Figure 2.15: Boxplots of all scoring functions result from the four deep learning models, (**a**) DNN, (**b**) Bi-GRU, (**c**) CNN1D, and (**d**) Bi-LSTM using a dynamically weighted loss function, and without the weight. The asterisk on the top of each boxplot denotes the $p$-value where "***" $< 0.001$, "**" $< 0.01$, "*" $< 0.05$.

171 features from sensors and the name for the features have been anonymised for proprietary reasons. The training data consists of 60,000 instances of which 59,000 belong to the negative class (no fault) and 1000 to the positive class (faulty). As for the testing data, it consists of 16,000 instances of which 15,625 belong to the no fault class and 375 to the fault class. The number of instances between the positive class and negative class is highly imbalanced as shown in Table 2.6.

Table 2.6: Number of instances and percentage of minority class in training and testing data of air pressure system (APS) failure dataset.

| Data | Number of Positive Instances | Number of Negative Instances | Percentage of the Minority Class |
|---|---|---|---|
| Training | 1000 | 59,000 | 1.67% |
| Testing | 375 | 16,000 | 2.34% |

### Data Preprocessing

The APS dataset contains 170 features and a binary class (True or False) as labels. The missing data are imputed using the K-Nearest Neighbour (KNN) [74] algorithm. Furthermore, the Synthetic Minority Over-Sampling Technique (SMOTE) is used to re-balance the training set as it is highly imbalanced as shown in Table 2.6. SMOTE is a way of increasing the minority class without directly duplicating instances of the minority class. Instead, new instances are synthesised within the clusters of minority classes. The reason this dataset is balanced for this experiment is because the goal is to investigate the effect of a dynamically weighted loss function on instances that are difficult-to-learn. Therefore, the balanced data is tested on DL models using normal and dynamically weighted loss functions. Further study is required to study the effect of dynamically weighted loss function on a highly imbalanced dataset.

The new instances are not just copies of existing minority cases; instead, the algorithm takes samples of the feature space for each target class and its nearest neighbours, and generates new examples that combine features of the target case with features of its neighbours. This approach increases the features available to each class and makes the samples more general.

### Deep Learning Architectures Investigated

For consistency, the same DL architectures listed in Section 2.5.1 are used to test the FL shown in Equation (2.36). Their respective hyperparameters are listed in Table 2.7. Furthermore, the strategy adopted for mitigation of overfitting is the same as the technique described in Section 2.6.1 using a combination of L2 regulariser and drop rate of 0.5. In addition, the $\gamma$ and $\alpha$ set for FL are 5 and 0.75 respectively. The $\gamma$ and $\alpha$ value were experimented using a combination of $\gamma = [1, 2, 3, 4, 5]$ and $\alpha = [0.25, 0.5, 0.75, 1.0]$. The results are shown in Figure 2.17 using a boxplot with different values of alpha and gamma. Additionally, it is ran six times to obtain the distribution. By using analysis of variance, it is found that the cost calculated using Equation (2.37) using different combinations of $\gamma$ and $\alpha$ were not statistically significant different as it has the F-value of $F(19, 100) = 0.588$ and a $p$-value greater than 0.05 at $p = 0.907$.

### Focal Loss Function

For many fault detection tasks, the goal is to predict the faulty and non-faulty condition given the sensors value. In essence, this is a binary classification problem.

A deep learning model typically produces a probability value for each class using the softmax activation function, and the loss is calculated using the CE loss function. Softmax activation function normalises the vector from a neural network prior to action layer ($Z$ in Figure 2.3) into a vector of probabilities proportional to the exponential of the given vector. For example, if the vector is $[1.3, 5.1, 2.2, 0.7, 1.1]$ it is converted to a vector of probabilities, $[0.02, 0.90, 0.05, 0.01, 0.02]$. Mathematically it is defined, as follows:

$$A(Z) = \frac{e^{Z_i}}{\sum_{j=1}^{K} e^{Z_j}} \tag{2.32}$$

Where $Z$ is the output vector before the activation function and $K$ is the number of elements in the vector. CE loss is a measure of the difference between two probability distributions. The first probability distribution is the actual class where the known class label has a probability of 1.0 and there is a probability of 0.0 for all other class labels. Subsequently, the second probability distribution is the predicted probability for each class. The CE loss function for binary classification can be represented mathematically as follows:

$$CE(y, p) = -(y * log(p) + (1 - y) * log(1 - p)) \tag{2.33}$$

where $p$ is the DL model probabilistic output that ranges from $[0, 1]$ and $y$ is the

ground truth class either 0 or 1. Equation (2.33) can be simplified to the form:

$$CE(y,p) = \begin{cases} -log(p) & \text{if } y = 1 \\ -log(1-p) & \text{otherwise.} \end{cases} \tag{2.34}$$

To further simplify the CE loss function, $p_o$ can be defined as:

$$p_o = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise.} \end{cases} \tag{2.35}$$

Therefore, $CE(p_o) = -log(p_o)$. Subsequently, a weighting factor term is added to convert the CE loss function to FL,

$$FL(p_o) = -\alpha_o(1-p_0)^{\gamma} log(p_o) \tag{2.36}$$

where $\alpha$ is a value between $[0,1]$ and $\gamma \geq 0$. Both $\alpha$ and $\gamma$ are tunable hyperparameters to optimise the performance of deep learning models. The $\alpha$ is a weighting parameter used to control the class imbalance problem. Additionally, the $\gamma$ is a focusing parameter that controls the loss. Larger values of $\gamma$ correspond to larger losses for badly learned instances. The data flow for the FL is the same as the proposed loss function in Section 2.3 and is it summarised in Figure 2.16.

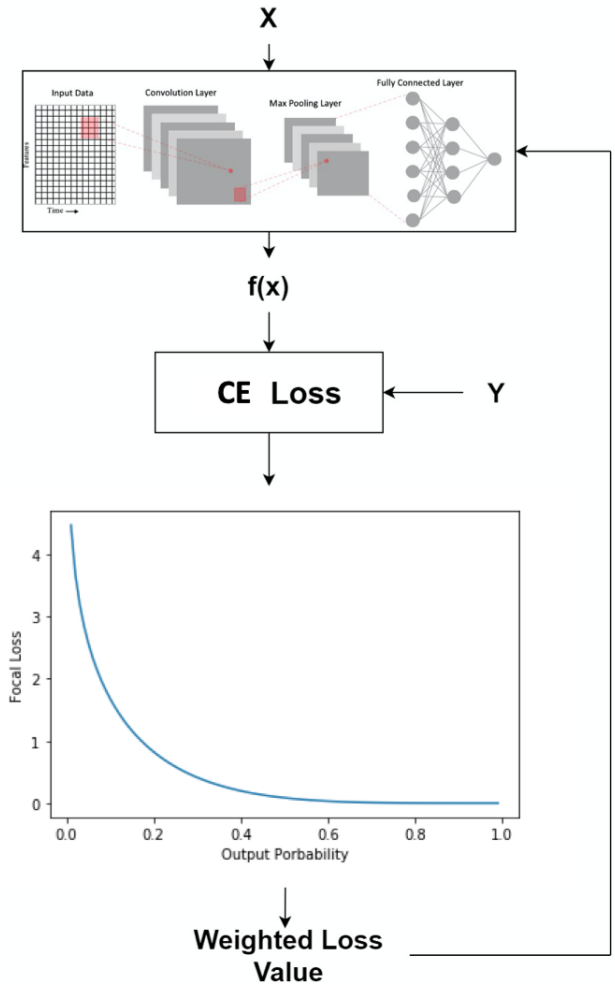



Figure 2.16: The output $f(x;\theta)$ from the deep learning model and the ground truth $Y$ are used to calculate the cross entropy (CE) loss for one instance. The CE is then combined with the weighted function to produce the weight that will be used to dynamically adjust the loss function.

### Evaluation Metrics

The authors of the APS dataset from Scania published a cost-metric of misclassification as an evaluation metric. Binary classification has two faults: (1) False positive

and (2) false negative. Each misclassification has a cost associated with it. In the context of the CBM, a false negative outcome has a more severe consequence compared to false positive outcome, and leads to an asymmetry in cost. A cost value of 10 and 500 are assigned to the false positive and false negative outcomes respectively to signify the asymmetry in cost. The cost value for the false positive and negative are specified by the data owner. The origin for the specific value of 10 and 500 are not explained. The total cost can be summarised as follows in Equation (2.37) and Figure 2.18:

$$\text{Total Cost} = (\text{Cost1} * \text{False positives}) + (\text{Cost2} * \text{False negatives}) \tag{2.37}$$

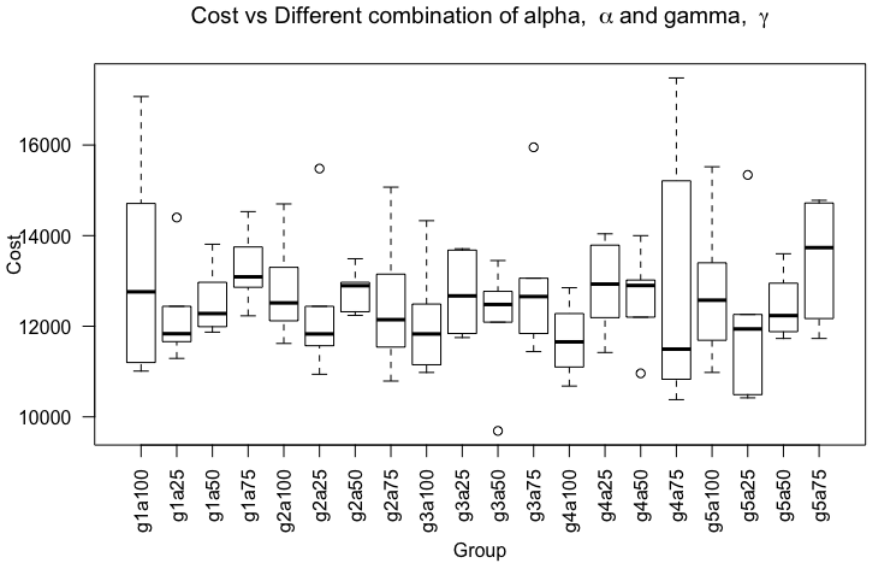



Figure 2.17: Boxplot of final cost using a combination of gamma values, [1, 2, 3, 4, 5] and alpha value, [0.25, 0.5, 0.75, 1.0]. The x-axis are denoted by the combination of alpha and gamma. For instance, 'g1a100' represents gamma value of 1 and alpha of 1.0.

Table 2.7: Hyperparameters of all models used to test the focal loss function presented in Section 2.6.2.

| **Deep Learning Architecture** | **Hyperparameters** |
|---|---|
| Bi-LSTM | Number of layers: 2 |
| | Layer 1 units: 32 |
| | Layer 2 units: 16 |
| | Activation function: ReLU |
| DNN | Number of layers: 2 |
| | Layer 1 units: 64 |
| | Layer 2 units: 64 |
| | Activation function: Sigmoid |
| CNN1D | Number of layers: 1 |
| | Layer 1 units: 30 |
| | Activation function: ReLU |
| | Filter size: $10 \times 1$ |
| Bi-GRU | Number of layers: 2 |
| | Layer 1 units: 32 |
| | Layer 2 units: 16 |
| | Activation function: ReLU |

The goal this fault classification task is to minimise the cost. A large percentage of the cost factor comes from the false negative classification. Additionally, metrics such as False Negative Rate (FNR), False Omission Rate (FOR), and recall are also used. The formulas for FNR, FOR and Recall are as follows:

$$\text{False Negative Rate} = \frac{\text{False Negative}}{\text{True Positive} + \text{False Negative}} \tag{2.38}$$

$$\text{False Omission Rate} = \frac{\text{False Negative}}{\text{True Negative} + \text{False Negative}} \tag{2.39}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{2.40}$$

Finally, a Precision-Recall (PR) curve is added to study the relationship between precision and recall. Precision is the measurement for the false positive rate while recall is the rate of true positive against false negative. Another option is the Receiver Operating Characteristic (ROC) curve. ROC curve is created by plotting the true positive rate against the false positive rate. A PR curve is employed here as opposed to a ROC curve as the latter can be easily misinterpreted in highly imbalanced datasets [75].

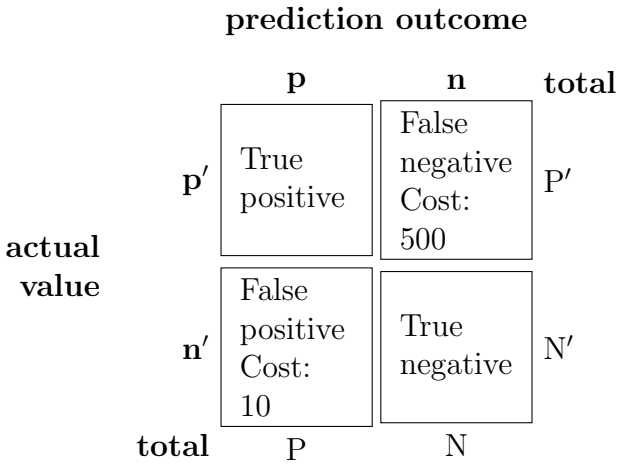



Figure 2.18: A confusion matrix with the associated cost of each fault. A confusion matrix tabulates the performance of a classification model. A true positive and true negative are correct classification therefore, there are no cost associated to it. Whereas false positive and false negative receive a cost of 10 and 500 respectively. The $p$ and $n$ represents positive and negative class while $P$ and $N$ represents the total positive and negative class. The actual class is denoted by an apostrophe, $'$.

### Results and Discussion

Table 2.8 shows the cost, FNR, FOR, and recall for the DL architectures Bi-LSTM, DNN, CNN1D, and Bi-GRU using CE and FL. Bi-LSTM, DNN, CNN1D, and Bi-

GRU with FL showed significant improvement across the cost, FNR, FOR, and recall metrics. CNN1D with FL achieved the lowest cost of 12,580 while Bi-GRU with CE loss achieved the highest cost of 35,480. Furthermore, when FL is used as the choice of loss function the cost metric improved by an average of 31.5% across the tested models.

The results of cost metric were plotted to show the distribution of output across 20 experimental runs as shown in Figure 2.19d. The number of "*" in Figure 2.19d represents the $p$-value where "***" $< 0.001$, "**" $< 0.01$, "*" $< 0.05$. The boxplots show that using DL models with FL as the loss function resulted in improvements that were statistically significant. DNN, CNN1D, and Bi-LSTM had $p$-values of $<0.001$ while Bi-GRU had a $p$-value of $<0.01$. In addition, the anomalies within the experimental runs for each DL models are shown in the boxplots as black dot.

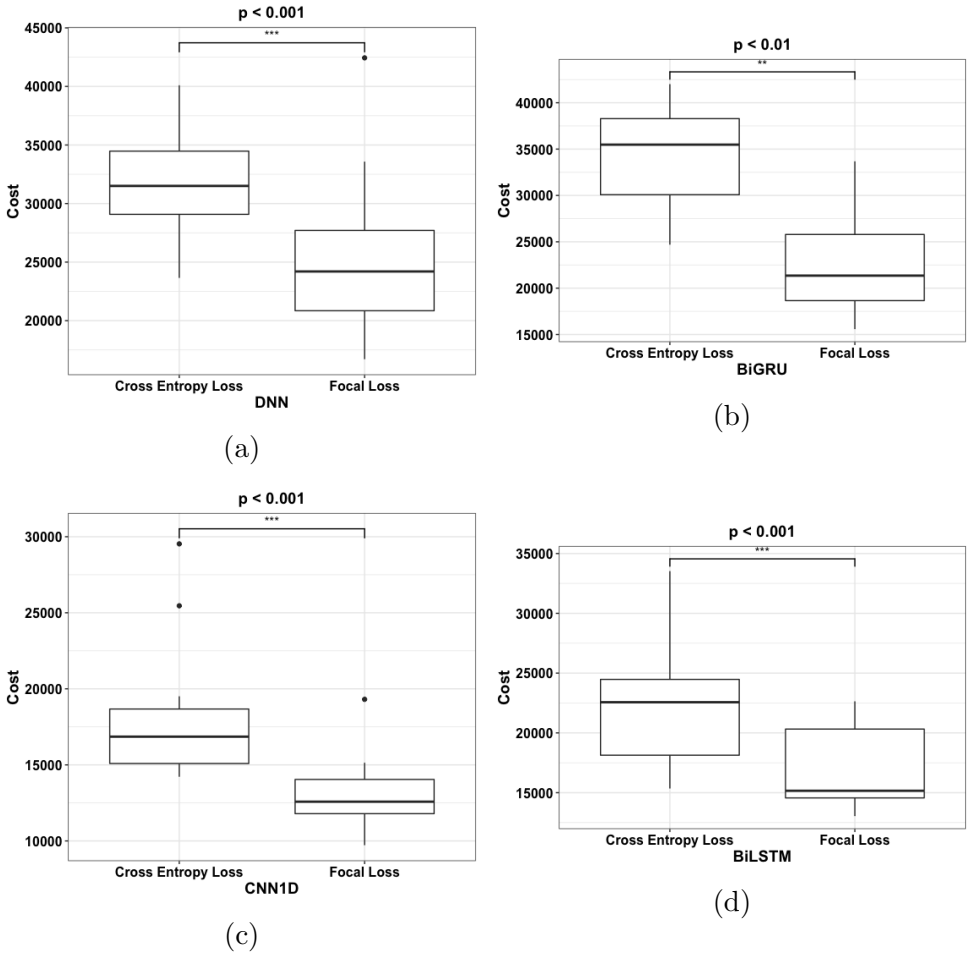



Figure 2.19: Boxplots of Costs result from using (**a**) DNN, (**b**) BiGRU, (**c**) CNN1D, and (**d**) BiLSTM with CE and FL respectively. The asterisk on the top of each boxplot denotes the $p$-value where "***" $< 0.001$, "**" $< 0.01$, "*" $< 0.05$.

Figure 2.20 displays the PR Curve for both FL and CE used with DNN, Bi-GRU, CNN1D, and Bi-LSTM. From the plot it can be observed that DNN's PR curve and Area Under the Curve (AUC) for FL and CE are similar. Bi-GRU with FL has a higher AUC compared to Bi-GRU with CE and overall achieved higher precision for a given recall. CNN1D and Bi-LSTM produced lower AUCs when FL

is used. CNN1D and Bi-LSTM both had a lower precision to achieve the same recall, with CNN1D having lower precision compared to Bi-LSTM. This is caused by the overwhelming false positive prediction to achieve the low false negative count. However, as mentioned in Section 2.6.2 the cost of a false positive is significantly lower than a false negative, at a ratio of 10:500. Therefore, when the actual cost is accounted for, as shown in Table 2.8 DL models with FL still outperformed CE in all cases.

Table 2.8: Results of cost, false negative rate, false omission rate, and recall using Bi-LSTM, DNN, CNN1D, and Bi-GRU with CE and FL. Blue colored text indicates improved performance. The values in this table are the median values across 20 experimental runs.

| **Deep Learning Architectures** | **Cost** | **False Negative Rate** | **False Omission Rate** | **Recall** |
|---|---|---|---|---|
| Bidirectional LSTM | 22,565 | 0.101 | 0.00248 | 0.898 |
| Bidirectional LSTM + FL | 15,160 | 0.045 | 0.00113 | 0.954 |
| | −32.8% | −55.4% | −54.4% | +6.2% |
| DNN | 31,505 | 0.156 | 0.00378 | 0.844 |
| DNN + FL | 24,200 | 0.112 | 0.00273 | 0.888 |
| | −28.2% | −39.3% | −27.8% | +5.0% |
| CNN1D | 16,855 | 0.067 | 0.00164 | 0.933 |
| CNN1D + FL | 12,580 | 0.012 | 0.00030 | 0.988 |
| | −25.4% | −82.1% | −81.7% | +5.9% |
| Bidirectional GRU | 35,480 | 0.177 | 0.00429 | 0.822 |
| Bidirectional GRU + FL | 21,350 | 0.074 | 0.00187 | 0.925 |
| | −39.8% | −58.1% | −56.4% | +12.5% |

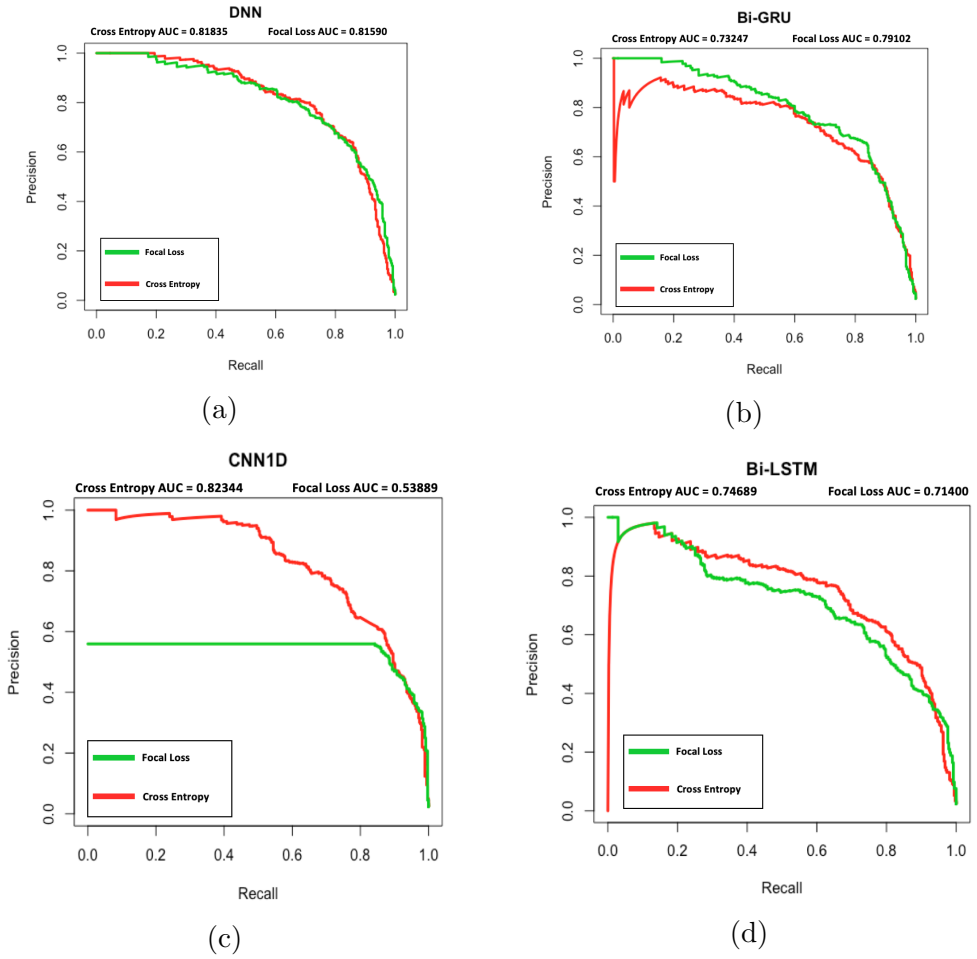



Figure 2.20: PR Curve for (**a**) DNN, (**b**) Bi-GRU, (**c**) CNN1D, and (**d**) Bi-LSTM using focal loss (Green line) vs. cross entropy loss (Red line). The AUC of PR curves are included at the top of each plot for each loss function.

The two case studies on weighted loss functions demonstrates that the CBM of a gas turbine engine and APS system are improved by using DL models with a dynamically weighted loss function that focused on instances that are poorly learned during the training process that can be thought of as a way to implicitly embed knowledge. The proposed loss function and FL are aimed at increasing prognostics

and diagnostics accuracy by improving on the existing loss function while keeping the DL architecture unchanged. Improvements are observed for the RUL prediction accuracy on the CMAPSS dataset and fault detection on the APS failure dataset classification performance using four different DL architectures, e.g., DNN, CNN1D, Bi-GRU, and Bi-LSTM, when the dynamically weighted loss function is used. Subsequently, the results are validated by performing a Mann–Whitney–Wilcoxon non-parametric statistical significance test, which showed the main evaluation metric, being function for case study 1 and cost for case study 2. All deep learning architectures tested achieved statistically significant improvement ($p < 0.05$) when the dynamically weighted loss function is employed.

### 2.6.3 Case Study on Asymmetric Loss Function: Remaining Useful Life Prediction of Gas Turbine Engine

This case study investigates the efficacy asymmetric loss functions on the problem of determining RUL for the CMAPSS dataset. The objective of this case study is to compare the asymmetric loss functions to traditional symmetric loss functions. The DL models used in Section 2.6.1 are re-optimised for the loss functions investigated.

#### Deep Learning Architectures

The following DL model architectures are employed to test the different loss functions in Section 2.4.1, (1) Bi-LSTM, (2) DNN, and (3) CNN1D. Their hyperparameters

are listed in Table 2.9. [1]. The asymmetric loss functions tested are MLSE-MSE, LIN-MSE, LIN-LIN, and QUAD-QUAD.

Table 2.9: Hyperparameters of all models used to test the asymmetric loss functions

| Deep Learning Architecture | Hyperparameters |
|---|---|
| Bi-LSTM | Number of layers: 3 |
| | LSTM Layer 1 units: 50 |
| | LSTM Layer 2 units: 25 |
| | Dense Layer 1 units: 50 |
| | Activation function: ReLU |
| DNN | Number of layers: 6 |
| | Dense Layer 1 units: 100 |
| | Dense Layer 2 units: 250 |
| | Dense Layer 3 units: 100 |
| | Dense Layer 4 units: 250 |
| | Dense Layer 5 units: 12 |
| | Dense Layer 6 units: 6 |
| | Activation function: ReLU |
| CNN1D | Number of layers: 5 |
| | Conv1D Layer 1: [Filter: 60, Kernel Size: 2] |
| | Conv1D Layer 2: [Filter: 60, Kernel Size: 2] |
| | MaxPooling 1D Layer |
| | Dense Layer 1 units: 100 |

In addition, L2 regulariser is added to the dense layers of all models shown in Table 2.9 to assist in the reduction of overfitting. The L2 regulariser reduces the

[1]The CMAPSS data and code for asymmetric loss function can be accessed on https://github.com/divishrengasamy/Asymmetric-Loss-Functions-for-Deep-Learning-Early-Predictions-of-Remaining-Useful-Life-in-Aerospace-/

chances of overfitting by limiting the complexity of the network through the penalisation of larger weights, thus, keeping the weights of the network smaller. Similarly, a dropout [44] rate of 0.2 is also added to all dense layers to mitigate overfitting. Dropout is a technique for regularising the network by randomly setting the output of units to zero.

### Results and Discussions

Table 2.10: Final score using Bi-LSTM, DNN, and CNN1D with symmetrical loss functions: MSE, and MAE and asymmetrical loss functions: MLSE-MSE, LIN-MSE, LIN-LIN, and QUAD-QUAD. The gray numbers in the square brackets represent the 95% confidence interval and the bolded numbers highlight the best score for each DL model.

| Loss Functions | Score [95% Confidence Interval] | | |
|---|---|---|---|
| | Bi-LSTM | DNN | CNN1D |
| MSE | 1554.4 [1039.5, 2069.2] | 4289.9 [3810.8, 4768.9] | 1044.7 [690.59, 1398.9] |
| MAE | 1162.4 [721.84, 1603.0] | 2001.1 [1263.9, 2738.4] | 1365.6 [975.42, 1755.9] |
| MLSE-MSE | 3037.1 [1688.4, 4385.8] | 5049.9 [4694.3, 5405.5] | 3761.8 [2741.4, 4782.3] |
| 0.1LIN-MSE | 2847.4 [1963.1, 3731.7] | 4509.0 [4026.9, 4991.1] | 3653.1 [2853.2, 4453.0] |
| 0.5LIN-MSE | 2965.0 [0, 6776.0] | 3227.3 [2820.1, 3634.5] | 3821.2 [430.62, 7211.9] |
| 1.0LIN-MSE | 2301.5 [1902.3, 2700.6] | 3255.6 [2941.7, 3569.6] | 2132.3 [600.31, 3664.2] |
| 1.0LIN-2.0LIN | **1027.1** [498.20, 1556.1] | **1725.0** [1401.7, 2048.4] | 1357.3 [814.79, 1899.8] |
| 2.0LIN-4.0LIN | 1670.5 [1596.6, 1744.3] | 2034.9 [1942.3, 2127.5] | 1392.1 [81.903, 2702.4] |
| 2.0LIN-6.0LIN | 2125.0 [1908.2, 2341.7] | 1769.0 [1447.5, 2090.5] | 1665.5 [1550.8, 1780.1] |
| QUAD-QUAD | 1647.3 [993.25, 2301.5] | 2203.8 [0, 5597.5] | **803.90** [670.78, 937.03] |

Table 2.10 shows the comparison of RUL score between Bi-LSTM, DNN, and CNN1D using MSE, MAE, MLSE-MSE, LIN-MSE, LIN-LIN, and QUAD-QUAD loss functions. The results reveal that the best overall score of 1027.1 and 1725.0 for Bi-LSTM and DNN is obtained when using asymmetrical loss function LIN-LIN with parameters ($a = 1.0, b = 2.0$). CNN1D has the best score of 803.9 using the QUAD-QUAD loss function. This shows that DL models are able to take advantage of the bias created by asymmetric loss functions to achieve early prediction and produce the lowest score for this case study. The asymmetric loss function shifted the learned representation of the data and also the loss landscape to adapt to the embedded knowledge. It can also be observed that symmetrical loss achieves a better score than asymmetrical loss in many results from the table. For example, Bi-LSTM and DNN with MAE achieve a better score than all asymmetrical loss functions, except for LIN-LIN with parameters ($a = 1.0, b = 2.0$). Similarly for CNN1D, MSE achieves the best score after QUAD-QUAD loss function.

In addition, $a = [0.6, 0.7, 0.8, 0.9]$ is used for the QUAD-QUAD loss functions and it is found that it does not converge for Bi-LSTM unless the architecture is modified. A possible reason for the non-convergence of Bi-LSTM could be because of the sensitivity to the larger loss due to weights of the asymmetric loss. The large weights multiple prevents the loss from decreasing. Another possibility is due to mis-parameterisation of the architecture and can be fixed by expanding the search space of hyperparameters. Therefore, the results are excluded to keep the architecture unchanged throughout the experimentation process. Furthermore, recurrent dropout [76] is experimented for Bi-LSTM but found no improvement in the model's

performance. The scores within the grey square bracket in Table 2.10 represent the lower and upper bound of the 95% confidence interval. The limit of the lower bound is set to zero as a negative is not possible, as shown in Fig 2.14. For example, the confidence lower bound of Bi-LSTM with 0.5LIN-MSE and DNN with QUAD-QUAD loss are set to zero due to negative lower bound scores.

In all asymmetric loss functions configuration tested 1.0LIN-2.0LIN and QUAD-QUAD consistently has better scores compared to others. The reason that 1.0LIN-2.0LIN and QUAD-QUAD does well is because it is similar to MAE and MSE in terms of loss landscape with an additional a small bias in favour for early predictions. The small bias does not change the learned representation and loss landscape drastically and ensure that the loss reaches a minima. This shows that using asymmetrical loss require additional parameters search and optimisation to obtain an optimal score. The results are also highly sensitive to the changes in the parameters of asymmetric loss functions. The reason for the high score from using some asymmetrical loss functions is that it is caused by the strong bias. The loss multiplier can increase or decrease the loss too much causing instability during training. By changing the loss landscape drastically while keeping all other model parameters the same the training has a reduced chance of the loss reaching the minima. A possible solution and future work is to extend the asymmetric loss function to be a learned loss function where the parameters of the asymmetric loss functions are learned and optimised. This ensures that the magnitude of its parameter does not cause instability during training. Furthermore, other parameters of the model such as the learning rate could be reduced the dampened the effect of strong bias resulted from

asymmetric loss function.

## 2.7 Summary

This chapter proposes a regression-based dynamically weighted loss function and also introduced FL, a classification-based dynamically weighted loss function to predictive maintenance. The objectives are to develop and test supervised DL models with a new type of knowledge embedded loss functions to improve predictive capabilities for sensor-based data by favouring early prediction in RUL and subsequently asses the advantage of knowledge embedded loss functions on RUL of gas turbine engine and fault detection in air pressure systems. Dynamically weighted loss functions are loss functions that implicitly embed knowledge to focus on difficult-to-learn instances during the model learning process. The two dynamically weighted loss functions were tested on RUL for gas turbine engine and fault classification of APS. The dynamically weighted loss functions are shown to improve prediction accuracy of DL-based models when compared to traditional loss functions for the two aforementioned case studies. Additionally, an asymmetric loss function that is specially embedded with knowledge of early prediction is more favourable than late prediction in RUL task is proposed. In the best case scenario, the asymmetric loss function outperforms standard loss functions when tested on the gas turbine engine RUL dataset. However, in most cases traditional loss functions still outperforms the asymmetric loss. While asymmetric loss function is able to outperform traditional loss functions, its parameter tuning is sensitive and require careful adjustment to achieve optimal results. Furthermore, a

more specific loss function specific to gas turbine engine can potentially yield more accurate RUL prediction results.

So far, implicitly and explicitly embedding knowledge into loss functions have been investigated and they both showed improvement in the prediction accuracy for prognostics and diagnostics task over generic loss functions. However predictive capability alone does not mean a ML model is suited to be deployed in production. In addition to embedding domain knowledge to aid the learning process, the knowledge learned by the models must also be extracted to ensure that models are explainable. In the following chapter, a novel way to accurately extract knowledge learned by the ML model is proposed and investigated.

# Chapter 3

# Extracting Knowledge from Machine Learning Models in Condition-based Maintenance to Improve Accuracy

## 3.1 Introduction

When machine learning supports decision-making in safety-critical systems, such as predictive maintenance, it is essential to verify and to understand why a particular output is produced. One way to achieve this is by extracting the knowledge learnt by the models about the task it is trained to achieve. Historically, however, many complex ML models, especially those involving neural networks, are viewed as 'black

boxes', where little is known about how the decision-making process occurs. The lack of adequate interpretability and verification of ML models [77, 78] has therefore prevented an even wider adoption and integration of those approaches in high-integrity systems. For domains where mistakes are unacceptable due to safety, security and economic issues they may cause, the need to accurately interpret the predictions and inference of ML models becomes imperative.

The recent rise in complexity of ML architectures has made it even more challenging to explain their outputs. Although there is a general agreement about the safety and ethical needs for interpreting ML outputs [78, 79], there is no consensus on how this challenge can be addressed. On the one hand, there are advocates for the development of models that are themselves interpretable, rather than putting the effort later in making black-box models explainable [80]. The argument is that for critical decision making, explanations of the black-box models are often unreliable, can be misleading and therefore unsafe. Conversely, other researchers have focused their efforts on explaining complex ML models; and significant advances have been achieved [81, 82, 83].

As explainable methods emerge, the area of predictive maintenance has also adopted them to understand the outputs of ML and DL models. Carletti *et al.* [84], for instance, analysed the important features that contribute to anomalies in pressure in manufacturing of refrigerators. Kumar and Hati [85] employ Deep CNN to identify the types of fault that occur in squirrel cage induction motors images. They use SHapley Additive exPlanations (SHAP) to highlight the area of the image that caused the CNN model to predict a fault. For RUL, Hong *et al.* [86] use SHAP

to identify which sensors contributes to the model prediction on gas turbine engine RUL. Szelazek *et al.* [87] employ SHAP to predict time to fault in a hot rolling of steel industrial process. While SHAP and Permutation Importance (PI) are useful methods in quantifying FI, they only provides overall effect of features on the model output. Some situations require fine-grained explanation on how each data instances are influencing the model output. Local interpretable model-agnostic explanations (LIME) [88] is an model agnostic explainer that explains how an input affects the model output. Serradilla *et al.* [89] employs a combination of both SHAP and LIME to quantify the importance of 97 sensor features in RUL prediction using GBT.

An essential approach to ML output elucidation is adopting post-training explanation given by FI estimates [90, 91]. Feature importance estimation quantifies how the features affects and contributes to the output. There are multiple methods for calculating FI, and they do not necessarily agree on how a feature relevance is quantified. It is not easy, therefore, to validate estimated FI unless the ground truth is known. Furthermore, there is no consensus regarding which is the best method or metric for FI calculation.

The lack of consensus of current approaches in determining the importance of data attributes for ML decision making is a problem for safety-critical systems such as predictive maintenance, as the explanation offered for the outcomes obtained is likely to be unreliable. There is therefore the need for more reliable and accurate ways of establishing FI. One possible strategy is to combine the results of multiple FI quantifiers, as a way to reduce the variance of estimates, leading to more a more robust and trustworthy interpretation of the contribution of each feature to the final

ML model prediction. In this chapter, a general, adaptable and extensible decision fusion framework is proposed named Multi-Method Ensemble (MME) for crisp FI fusion with the aim of reducing variance in current FI estimates. The objective is to develop and test the MME framework for decision fusion that is more accurate in FI estimates compared than the existing FI method. An additional objective is to develop an understanding of how MME behaves under varied synthetic and real-word data conditions. MME merges results from multiple machine learning models with different FI calculation approaches using data bootstrapping and decision fusion techniques. The main process of the MME are: (i) optimised ML algorithms, (ii) a set of FI coefficients from optimised ML algorithms and FI calculation techniques, (iii) a set of aggregated importance coefficients produced by multiple decision fusion techniques. The proposed MME framework is tested on a multitude of synthetic data that resembles real-world data. The primary evaluation on the framework is conducted on synthetic data because it provides quantifiable ground truth on the features as opposed to real-world data where the FI of its features are typically qualitative, making it difficult to empirically verify FI accuracy. Nonetheless, the MME framework is also conducted on a real-world creep prediction task.

## 3.2 Background on Ensemble Feature Importance

Early approaches to FI quantification utilise interpretable models, such as linear regression [92] and logistic regression [93]; or ensembles, such as generalised linear models [94], and DT [40] to determine how each feature contributes to the model's

output [95]. As data and data problems become more challenging and convoluted, simpler and interpretable models need to be replaced by complex ML solutions. For those, the ability to interpret predictions without the use of additional tools becomes far more difficult. Model-agnostic interpretation methods are commonly used strategies to help determining the FI from complex ML models. They are a class of techniques that determine FI, while treating models as black-box functions. In this section a review of the current literature on ensemble FI, including identifying gaps in the literature, the basic concepts, and rationale for choosing model-agnostic approaches in the proposed framework experiments.

### 3.2.1 Related Work

One of the earliest ensemble techniques to calculate FI is RF, proposed by Breiman [40]. RF is a ML model that forms an ensemble of decision trees via random subspace methods [96]. Besides prediction, RF computes the overall FI by averaging those determined by each decision tree in the ensemble. RF FI is quantified depending on how many times a feature branches out in the decision tree, based on the Gini impurity metric. Alternatively, decision trees also calculate FI as Mean Decrease Accuracy or more commonly known as PI by permutating the subset of features in each decision tree and calculating how much accuracy decreases as a consequence. Using the knowledge of ensembling FI from weak learners, De Bock *et* al. [97] proposed an ensemble learning based on Generalised Additive Models (GAM) to estimate FI and confidence interval of prediction output. Similarly to Bagging, the average of each weaker additive model generates the ensemble predictions. The FI scores are

generated using the following steps: (i) Generate output and calculate performance for individual predictions based on a specific performance metric; (ii) permute each feature and recalculate error for Out-Of-Bag (OOB) predictions; (iii) calculate the partial importance score based on OOB predictions; (iv) repeat step (i) to (iii) for each additive model and different forms of evaluation. The authors argue that the importance of each feature should be optimised according to the performance criteria most relevant to the feature in order to obtain the most accurate FI score. The GAM ensemble-based FI is subsequently applied to a case study that aims at identifying essential features in churn prediction to determine customers likely to stop paying for particular goods and services. To determine the ten most relevant features, the authors use receiver operating characteristic and top-decile lift. The authors observed that the sets of important features overlapped, but their rank order is different when using ROC and lift. The different rank orders show that FI is affected by the evaluation criteria. Both Breiman and De Bock *e*t al. use only a single ML model with one type of FI method to calculate the ensemble FI, which restricts the potential to improve accuracy and to reduce variance. To overcome this limitation, Zhai and Chen [98] employ a stacked ensemble model to forecast the daily average of air particle concentrations in China. The stacked ensemble consists of four different ML models, namely, Least Absolute Shrinkage and Selection Operator (LASSO) [99], Adaptive Boosting (AdaBoost) [100], XGBoost, and MLP with SVR as the meta-regressor. The authors use a combination of feature selection and model generated method to determine FI, which is determined from stability feature selections, XGBoost model and AdaBoost model. Their outputs are subsequently

averaged for the final ranking of features. AdaBoost and GBT use the Mean Decrease Impurity (MDI) based on the Gini importance; Sample frequency spectrum is based on maximum feature scores using bayesian information criterion [101]. The top ten features are selected for evaluation.

While Zhai and Chen used multiple ML models and one FI approach, according to the latest literature, there has not been further investigations to improve FI quantification using multiple models coupled with multiple FI methods. Furthermore, we have found no literature applying any form of ensemble FI to interpret the models' output and identifying key features that contribute to the outcome for decision making. Finally, to the best of our knowledge, there is little in-depth systematic investigation of how ensemble FI decision fusion works in the literature. Therefore, it is important that interpretability methods and ensemble FI fusion under different data conditions are investigated.

### 3.2.2 Feature Importance Calculation Approaches

#### Permutation Importance

PI measures FI by calculating the changes in model's error when a feature is replaced by a shuffled version of itself. The algorithm of how PI quantifies FI can be defined as follows:

The $shuffle$ function randomly changes the row position of the instances of the data. The magnitude of difference between $baseline\_performance$ and $error$ in Algorithm 1 signifies the importance of a feature. A feature has high importance if

**Algorithm 1:** Algorithms of permutation importance

```
Result: Permutation feature importance
Input: features, labels, Trained_Model
predicted_output = Trained_Model(features));
baseline_performance = Loss(Predicted_output, labels);
for i = 0; i < length(features); i++ do
    original_feature = features[i];
    shuffled_feature = shuffle(features[i]);
    features[i] = shuffled_feature;
    predicted_output = Trained_Model(features));
    error = Loss(Predicted_output, labels);
    feature_importance[i] = error - baseline_performance;
    features[i] = original_feature;
end
return feature_importance;
```

the performance of ML deviates significantly from the baseline after a shuffling; it therefore has low importance if the performance does not change significantly. PI can be run on train or test data but test data is usually chosen to avoid retraining of ML models to save computational overhead. If the computational cost is not an important factor to be considered, a drop-FI approach can be adopted to achieve greater accuracy. This is because in PI a shuffled feature might not differ much from the original order of the instances. For example, if a dataset has 5 instances and the original sequence of values for a feature is $[1, 1, 3, 4, 2]$, and after shuffling, the new order is $[1, 1, 4, 3, 2]$, there is not much difference between the pre-shuffled and shuffled instances, making it difficult to determine importance. In contrast, drop-FI excludes a feature, as opposite to performing its permutation leading to more accurate quantification of importance. However, as it requires the ML model to be retrained every time a different features are dropped, it is computationally expensive

for high dimensional data.

PI is a preferred alternative over MDI using Gini Importance. MDI tends to disproportionately increase the importance of continuous or high-cardinality categorical variables [102], leading to bias and unreliable FI measurements. PI is a model-agnostic approach to FI. When multiple ML models produce similar predictive accuracy, the features that affect the outcomes of each model vary across models. A disadvantage of PI is that it is not able to consider the synergy of multiple ML models, therefore making it unreliable as important features might be unaccounted for.

### Shapley Additive Explanations

SHAP is a ML interpretability method that uses Shapley values, a concept originally introduced by Lloyd Shapley [103] in game theory to solve the problem of establishing each player's contribution in cooperative games. Essentially, given a certain game scenario, the Shapley value is the average expected Marginal Contribution (MC) of a player after all possible combinations have been considered. For ML, SHAP determines the contribution of the available features of the model by assessing their every possible combination and quantifying their importance. The total possible combinations can be represented through a power set. For example, in the case of three features, $PowerSet\{x, y, z\}$ the power set is $\{\emptyset\}, \{x\}, \{y\}, \{z\}, \{x, y\}, \{x, z\}, \{y, z\}, \{x, y, z\}\}$. Furthermore, Figure 3.1 illustrates the relationships between the elements in the power set.

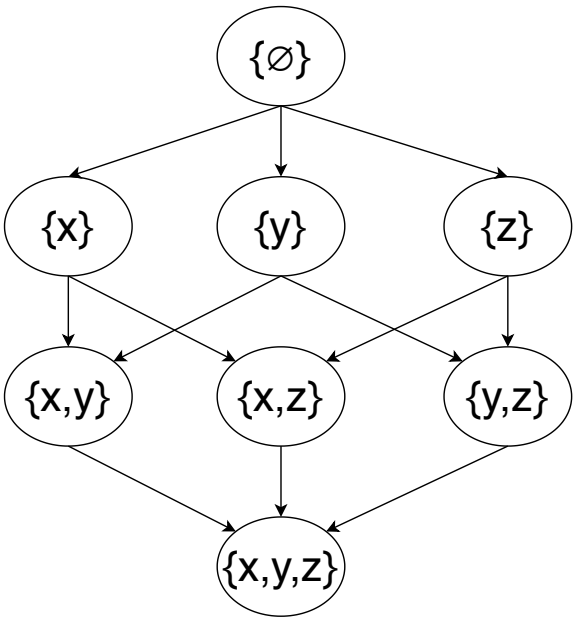



Figure 3.1: A graph representation of power set for features $\{x, y, z\}$. The $\emptyset$ symbol represents the null set which is the average of all outputs. Each vertex represents a possible combination of features, and the edge shows the addition of new features previously not included in previous group of features.

SHAP trains a ML model for each of the vertices shown in Figure 3.1. Therefore, there are $2^{\text{number of features}} = 2^3 = 8$ models trained to estimate the contribution of each feature. The number of models needed to estimate FI using SHAP increases exponentially with the number of features. However, there are tools, such as the *Python* library *SHAP* [104] to accelerate the process through approximations and sampling. The MC of a feature can be calculated by traversing the graph in Figure 3.1 and summing up the changes in output where the feature is previously absent from the combinations. For example, to calculate the contribution of feature $\{x\}$, the weighted average of the change in the output from $\{\emptyset\}$ to $\{x\}$, $\{y\}$ to $\{x, y\}$, $\{z\}$ to $\{x, z\}$, and $\{y, z\}$ to $\{x, y, z\}$. The MC of a feature $x$ going from $\{\emptyset\}$ to $\{x\}$ is as

follows:

$$MC_{x,(\emptyset,x)}(i_0) = Output_x(i_0) - Output_\emptyset(i_0) \tag{3.1}$$

where *Output* is the output of a ML model and following Equation 3.1 the SHAP value of feature $x$ of an instance, $i_0$ is calculated as follows:

$$\begin{aligned} SHAP_x(i_0) = w_1 * MC_{x,(\emptyset,x)}(i_0) + \\ w_2 * MC_{x,(x,y)}(i_0) + \\ w_3 * MC_{x,(x,z)}(i_0) + \\ w_4 * MC_{x,(x,y,z)}(i_0) + \end{aligned} \tag{3.2}$$

The process is repeated for each feature to obtain the FI. The weights $(w_1, w_2, w_3, w_4)$ in Equation 3.2 sum to 1. The weights are calculated by taking the reciprocals of the number of possible combinations of MC for each row in Figure 3.1. For example, the weight of $w_1$ is $\binom{3}{1}^{-1}$. To calculate the global importance of a feature, the absolute SHAP values are averaged across all instances.

### Integrated Gradients

There are several FI methods specifically designed for Deep Learning models, such as GradCAM [105], Guided GradCAM [105], Guided Backpropagation [106], DeepLift [107], and Integrated Gradients (IG) [108]. GradCAM and Guided GradCAM are only applicable to Convolutional Neural Networks (CNNs) and image analysis. Guided

Backpropagation fails to deliver reliable FI calculation, as their outputs remain invariant when the network is reparamaterised or when the test labels are randomly permutated [109]. Furthermore, a desirable property of gradient-based FI is *Completeness* which DeepLIFT and IG are designed to satisfy. *Completeness* property states that the sum of FI should be the difference between the model's output at a particular instance and the baseline. The baseline input is a vector of zero in the case of regression to ensure that the baseline prediction is neutral and it functions as a counterfactual. While DeepLIFT is originally designed to satisfy the *Completeness* property, it has since been shown that it fails to achieve that [110].

IG is a gradient-based method for FI. It determines the FI $A$ in deep learning models by calculating the change in output, $f(x)$ relative to the change in input $x$. Additionally, the change in input features is approximated using an information-less baseline, $b$. The FI is denoted by the difference between the characteristics of the deep learning model's output when features and baseline are used. The formula for FI using a baseline is as follows:

$$A_i^f(x, b) = f(x_i) - f(x[x_i = b_i]) \tag{3.3}$$

The individual feature is denoted by the subscript $i$. Equation 3.3 can be also written in the form of gradient-based importance as:

$$G_i^f(x, b) = (x_i - b_i)\frac{\partial f(x)}{\partial x_i} \tag{3.4}$$

IG obtains FI values by accumulating gradients of the features interpolated between

the baseline value and the input. To interpolate the two values, a constant, $\alpha$, with the value ranging from zero to one is used, as follows:

$$IG_i^f(x,b) = (x_i - b_i) \int_{\alpha=0}^{1} \frac{\partial f(b + \alpha(x-b))}{\partial x_i} d\alpha \tag{3.5}$$

Equation 3.5 is the final form of IG used to calculate FI in a deep learning model.

While the methods discussed in sections 3.2.2, 3.2.2 and 3.2.2 are currently used in predictive maintenance as discussed in Section 3.2.1, they have some disadvantages. For example, different FI techniques can produce different importance coefficients, often with diverse magnitudes for the same features and datasets, leading to uncertainty and inaccuracies in the FI output. Furthermore, when multiple learned models with predictive accuracy that are not statistically significantly different, each model can be employed to produce FI values and as a result leading to further increases in uncertainty as different learned models produces different FI values even when using the same FI techniques. The lack of consensus of current approaches in determining the importance of data attributes for ML decision making is a problem for safety-critical systems, as the explanations offered for the outcomes obtained are likely to be unreliable. In the context of predictive maintenance, the lack of clear interpretation of models output may affect ones ability to take appropriate maintenance action and to validate or 'challenge' the results. Our hypothesis is to combine the quantification of multiple FI quantifiers to reduce the variance in estimates, leading to a more robust and accurate interpretation of the contribution of each feature to the final ML model prediction. In the next section, we propose Multi-Method Ensemble (MME) for FI fusion with the objective of reducing the

variance in current FI estimates.

## 3.3 Multi-Method Feature Importance Ensemble Framework

This section introduces the proposed Multi-Method Feature Importance Ensemble (MME) framework where multiple ML models and global FI methods are ensembled to produce a final global FI quantification. As mentioned in Section 1.3, the hypothesis posits that an ensemble of multiple ML and DL models coupled with FI interpretation methods leads to more robust and reliable post-hoc FI measurement compared to using single models or single FI methods.

### 3.3.1 Ensemble Feature Importance

Figure 3.2 shows the MME framework in 4 different stages. On **Stage 1**, data undergoes pre-processing, such as transformation, noise reduction, feature extraction and feature selection. This stage is required, as it needs to ensure that the data has no inconsistencies and that the features used to train the ML models are orthogonal by removing noise and features that are collinear, which could negatively impact model's predictive performance. The preference for features with low correlation guarantees that the FI calculation does not attribute random values of importance because a set of features containing similar information can split the importance quantification. On **Stage 2** ML models are applied to the pre-processed data. Generally, only one model is applied to the dataset or multiple models are tested before selecting a final

best performining model. In the MME framework however, the goal is utilise multiple models with different methods to diversify the learned representations. Subsequently, model-agnostics FI methods are applied to the ML models (**Stage 3**). Each trained model has its FI extracted through the FI methods in the framework. The FI results are fused into a final FI decision in **Stage 4**, which is further introduced in the next section. The MME framework is adaptable and extensible.

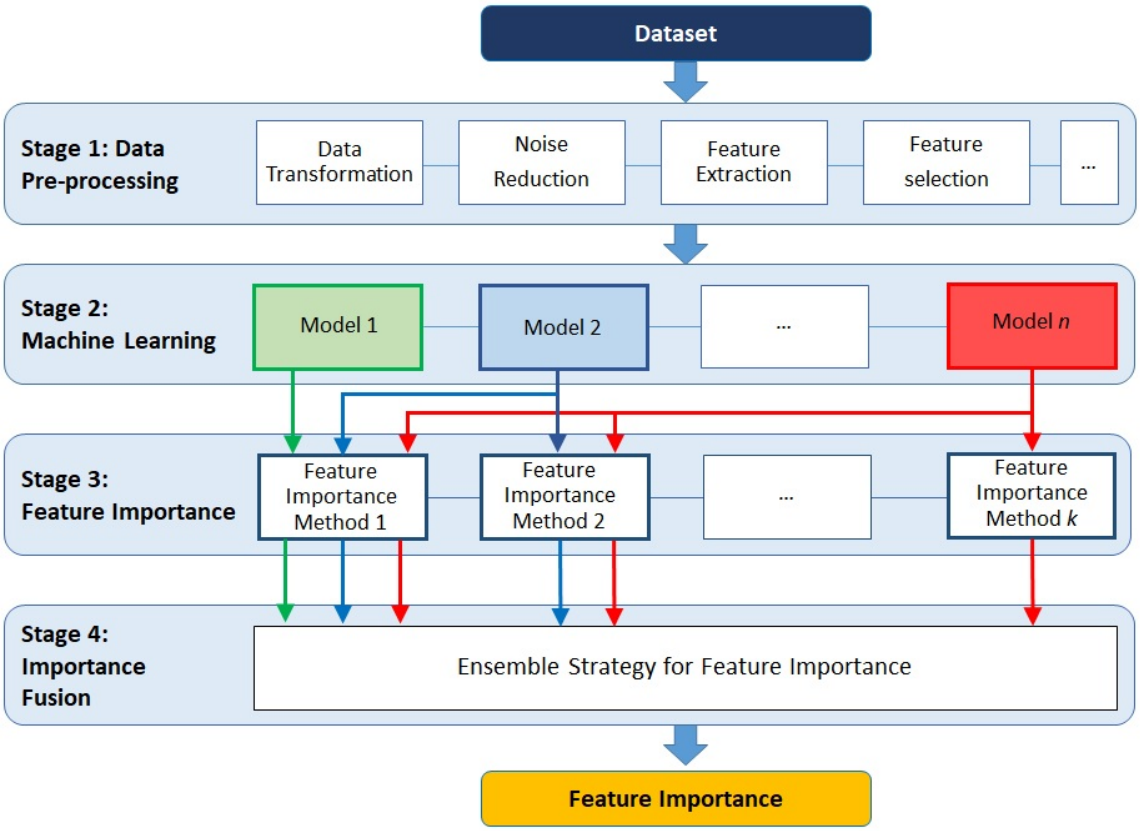



Figure 3.2: The four stages of the proposed FI fusion MME framework. The first stage pre-processes the data and in the second step trains the data on multiple ML models. The third step calculate FI from the each trained ML models using multiple FI methods. Finally, the fourth step fuses all FI generated from the third step using an ensemble strategy to generate the final FI values.

In **Stage 3** and **Stage 4** when multiple FI methods and ML models are employed in ensemble FI, we obtain several vectors, $Q$ of FI values, $A$ as follows: $Q = [A_1, A_2, ..., Ai]$ where $i$ is the number of features and there are one $Q$ for each ML model and FI technique combination. The FI vectors can be denoted as follows: $\overline{V} = [Q_1, Q_2, ..., Q_N]$ where $N$ is the sum of each model multiplied by the number of FI methods used. To reduce the noise and variance, we can take the average of all FI vectors, $\overline{V}$ which produces the variance, $\sigma^2/N$. As $N$ increases, the variance decreases. If an FI vector, $Q$ from a model or FI methods is substantially different from the other $Q$s, it is considered as an outlier. The variance and accuracy of final FI vector can be further improved if the outlier $Q$ are removed prior to taking the average.

### 3.3.2 Decision Fusion Strategies

Within the MME framework, the importance calculated is stored in a matrix, $\overline{V}$, and the decision fusion strategies are used to determine the final FI values from $\overline{V}$. The most common decision fusion strategy, as discussed in Section 3.2.1 is to use the average values. However, this is not the most suitable decision fusion strategy in cases where one or more of the FI approaches produce outliers compared to the majority of responses. In addition to the decision fusion using the average, median, mode, box-whiskers, majority vote, Modified Thompson Tau test, and RAnk correlation with majority voTE (RATE) are also investigated. For majority vote, each vector in the FI matrix have their features ranked based on their importance. Subsequently, the final FI is the average of the most common rank order for each feature. For example,

feature $X_i$ has a final rank vector of $[1, 1, 1, 2]$, where each rank $r_k$ is established by a different FI method $k$. The final FI value for feature $X_i$ is the average value from the three FI methods that ranked it as one. Modified Thompson Tau test is a statistical anomaly detection method using t-test to eliminate values that are above two standard deviations.

RATE is novel fusion approach proposed here that combines the statistical test, feature rank and majority vote. RATE combines the advantage of using a statistical approach to rank FI and anomaly removal with majority vote. Our hypothesis is that the usage of rank correlation which measures the strength and direction of association between FI vectors will improve the identification outliers within the matrix $\overline{V}$. The steps used in RATE are illustrated in Figure 3.3. The input to RATE is the FI matrix, $\overline{V}$. $\overline{V}$ is the matrix that has the individual FI from different models and it has the shape of $N * M$ where $N$ are the FI vectors from different importance calculation methods and $M$ is the number of features. Subsequently, the pairwise rank correlation between each FI vectors in the matrix $\overline{V}$ to obtain the rank and the general correlation coefficient values [111] is calculated. Using the rank and correlation value it can be determine whether the correlation between the vectors is statistically significant (p-value less than 0.05). The p-values are stored in a separate matrix that is converted to a truth table. If pairwise p-value is less than 0.05 it is given a value 'TRUE', otherwise,'FALSE'. Using the truth table along with majority vote, the FI vector that does not correlate with the majority of vectors is discarded. The remaining FI vectors are averaged as the final global FI.

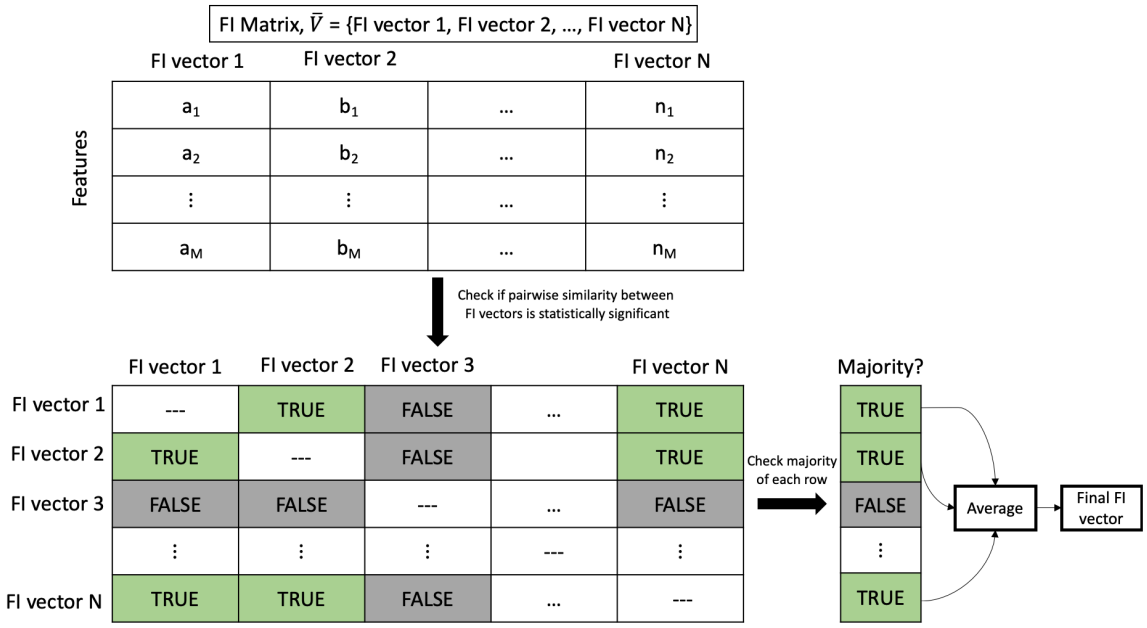



Figure 3.3: The working of RATE FI ensemble strategy. The FI vectors undergo a rank correlation pairwise comparison to determine if their similarity is statistically significant (p-value<0.05). A value of 'TRUE' is assigned if the two vectors are similar, otherwise, a 'FALSE' is assigned in a truth table. For each row of the truth table, majority voting (> 50% is TRUE) determines if the FI vector is accounted when calculating the final FI.

## 3.4 Experimental Design

To test the performance of the MME framework, two case studies are conducted on (1) synthetic data; and (2) a real-world dataset on creep rates in laser powder bed fusion. The synthetic data consists of data characteristics that mimics the real world such as different level of noise in data, number of features, and number of informative features. The hypothesis of the case studies is that the MME framework is able to produce more accurate FI quantification compared to single-method ensemble (SME) where only single FI method is used for multiple ML models. Mean is used

as the decision fusion strategy for SME. For the synthetic data its FI ground truth is generated to calculate the accuracy. For the real-world dataset the FI quantifications are validated by experts. The main objective of the second case study is to illustrate and validate the applicability of the MME framework in the real world. Both case studies starts with how the training and testing data is generated, preprocessed, and prepared. Subsequently, the appropriate ML methods and evaluation metrics are selected for the regression tasks. After the models are trained MME framework is employed to generate FI quantification. Finally, the results are presented and the findings are discussed.

### 3.4.1 Summary of Data Preprocessing

In Table 3.1 a summary of the data preprocessing employed is listed.

| **Task** | **Description** |
|---|---|
| Data split ratio | 80/10/10 (train/validation/test) |
| Data split method | Randomly split between train/validation/test or Leave-one-out |
| Scaling method | All features scaled between 0-1 |
| Feature Selection | None |
| Feature Extraction (Image) | Scikit-Image library |

Table 3.1: A table of data preprocessing methods for synthetic and creep rate prediction data.

## 3.5 Case Study 1: Synthetic Datasets

### 3.5.1 Data Generation

The data investigated are generated using *Python*'s *Scikit-learn* library [112] with different characteristics to mimic a variety of real-world regression scenarios. The reason for using synthetic data is such that the ground truth of FI for the datasets is controlled and easily obtained which is not the case for many real-world data. The features generated are random but it is well conditioned, centered, and Gaussian with unit variance by default. The correlation between features are also random. The parameters used to modulate the creation of the datasets are the standard deviation of the Gaussian distributed noise applied to the data, the number of features included and the percentage of informative features. Their values are shown in Table 3.2. Gaussian distributed noise with different standard deviations are added to the output as it has a more significant effect on prediction accuracy than that in the features [113]. Larger noise values push feature values further from the average, decreasing Signal/Noise ratios. Although noise increases ML models' estimated error [113, 114] studies investigating the relationship between data noise and FI error are scarce. Similarly, the impact of the number of features and how many features within the set are relevant to importance error is unknown. A combination of values from each parameter in the table forms a dataset, and permutations of those parameters form a total of 45 datasets. For each dataset, we conduct ten experimental runs to ensure that the results are stable and reliable.

The data preprocessing in this case study consists of scaling the input data to

Table 3.2: Parameters to generate the datasets used to test the MME framework.

| Parameters | Description | Parameters' value |
|---|---|---|
| Noise | Standard deviation of Gaussian noise applied to the output. | 0, 2, 4 |
| Informative level (%) | Percentage of informative features. Non-informative features do not contribute to the output. | 20, 40, 60, 80, 100 |
| Number of features | Total number of features used to generate output values. | 20, 60, 100 |

the range between zero and one. Scaling the data accelerates the learning process and reduces model error for neural networks [115]. Additionally, it allows for equal weighting of all features and therefore reduces bias during learning. Other common preprocessing steps such as anomaly removal and normalisation are not included as the data creation process is controlled.

### 3.5.2 Machine Learning Models

The ML approaches employed in the experiments are RF, GBT, SVM, and DNN as different ML methods are likely to have different levels of importance for each feature while producing good predictive accuracy to the output. The four models are applied to both SME and MME. The hyperparameters adopted are shown in Table 3.3. The models are optimised using random hyperparameters search [116]. The models are not optimised for individual datasets. The model's hyperparameters constant are

kept as they are a factor that affects FI accuracy. We therefore limit our objectives to investigating how data characteristics affect interpretability methods and how the appropriate decision fusion of different FI methods produces less biased estimates. We minimise overfitting in the deep learning model by adopting dropout [44] and L2 kernel regularisation [117]. The number of epochs for the training of deep learning model is not fixed as early stopping is used. If the loss remains constant for 10 epochs, the training is automatically stopped.

The FI methods employed by each ML models are listed in Table 3.4. For SHAP, we employ weighted k-means to summarise the data before estimating the values of SHAP. Each cluster is weighted by the number of points they represent. Using k-means to summarise the data has the advantage of lower computational cost but slightly decreasing the accuracy of SHAP values. However, we compare the SHAP values from data with and without k-means for several datasets and found the SHAP values to be almost identical.

### 3.5.3 Evaluation Metrics

To evaluate the performance of the ensemble FI and also the different ensemble strategies three different evaluation metrics are employed namely, MAE, RMSE, and $R^2$.

Table 3.3: Hyperparameters values for Random Forest, Gradient Boosted Trees, Support Vector Regressor, and Deep Neural Network for case study 1.

| Models | Hyperparameters | Values |
|---|---|---|
| Random Forest | Number of trees | 700 |
| | Maximum depth of trees | 7 levels |
| | Minimum samples before split | 2 |
| | Maximum features | $\sqrt{p}$ |
| | Bootstrap | True |
| Gradient Boosted Trees | Number of trees | 700 |
| | Learning rate | 0.1 |
| | Maximum depth of trees | 7 levels |
| | Loss function | Least square |
| | Maximum features | $\sqrt{p}$ |
| | Splitting criterion | Friedman MSE |
| Support Vector Regressor | Kernel | Linear |
| | Regularisation parameter | 2048 |
| | Gamma | 1e-7 |
| | Epsilon | 0.5 |
| Deep Neural Network | Number of layers | 8 |
| | Number of nodes for each layer | 64, 64, 32, 16, 8, 6, 4, 1 |
| | Activation function for each layer | ReLU, except for output is linear |
| | Loss function | MSE |
| | Optimiser | Rectified Adam with LookAhead |
| | Learning rate | 0.001 |
| | Kernel regulariser | L2 (0.001) |
| | Dropout | 0.2 |

Table 3.4: Interpretability methods employed for each ML model for FI decision fusion.

| Models | Interpretability methods |
|---|---|
| Random Forest | Permutation Importance, SHAP |
| Gradient Boosted Trees | Permutation Importance, SHAP |
| Support Vector Regressor | Permutation Importance, SHAP |
| Deep Neural Network | Permutation Importance, SHAP, and Integrated Gradient |

### 3.5.4 Results and Discussion

#### Overall Single-Method Ensemble vs Multi-Method Ensemble Framework

Figure 3.4 shows the average MAE results of all SME and the multiple decision fusion implementations of MME framework for the combined synthetic datasets investigated, including train-test split of the data. A comparison of the average results between SME and MME frameworks using the RMSE and $R^2$ metrics are shown in Appendix A.1.1 and Appendix A.1.2. The method with the smaller FI errors is our framework using majority vote for decision fusion, followed by the MME framework coupled with the mean and then RATE. SME such as using all ML models with SHAP, PI and IG produce the worst results. The circles and bars in the figure represent the FI errors on the training and test datasets, respectively. The FI errors on the training dataset are slightly lower than errors on the test dataset.

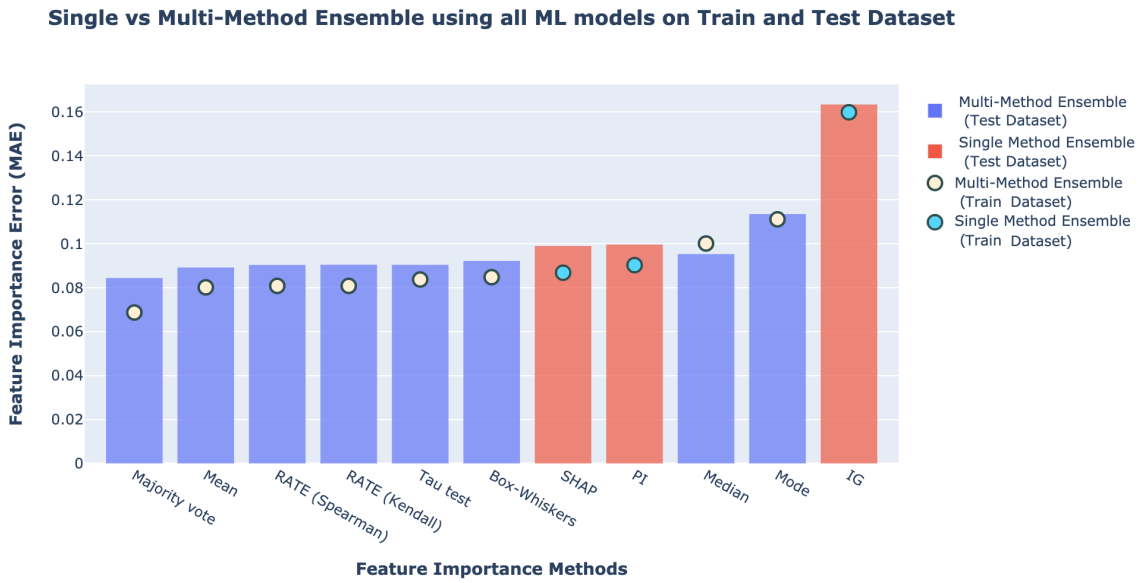



Figure 3.4: Average FI error between SME and MME framework with train and test dataset of the combined synthetic datasets with different varying level of noise, number of features, and informative features.

### Effect of Noise Level, Informative Level, and Number of Features

The results in the previous section shows the overall errors when the different data configurations such as different noise levels, number of features, and number of informative features are combined. In this subsection, the results shows the respective data configurations FI quantification accuracies. Figure 3.5 shows the FI results of SME and the MME framework with different fusion strategies averaged across three different noise levels in the data. Results on the effect of noise in data on ensemble FI using the RMSE and $R^2$ metrics are shown in Appendix A.1.3 and Appendix A.1.4 of the supplementary material. The best performing ensemble method averaged across all noise levels is MME framework using majority vote. MME framework that uses majority vote outperforms the best SME method, SHAP, by 14.2%. Table 3.5 and

Figure 3.6 show how the FI errors change for all SME and MME framework's decision fusion methods as the noise level of data increases. In Table 3.5 we observe that the MAE decreases marginally, from a noise level of 0 standard deviation to 2 standard deviations, and then it increases again 4 times the standard deviation. The addition of 2 standard deviation noise to the dataset improves the generalisation performance of ML models leading to lower errors [118]. However, the FI errors increase when the noise in the dataset reaches 4 times the standard deviation, indicating that the noise level has negatively impacted ML models performance. Overall, however, the noise levels have little effect on the FI errors. Results also reveal that MME framework with majority vote achieves the best FI estimates for the data.

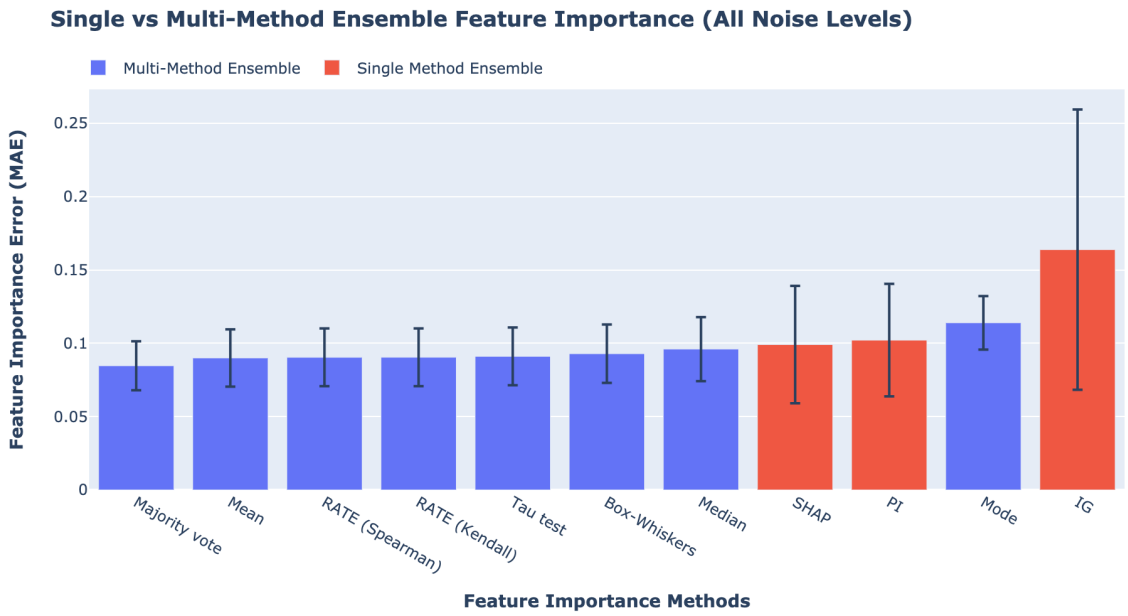



Figure 3.5: Effect of all noise levels for SME and MME framework with decision fusion methods

Table 3.5: FI MAE using SME and MME framework for different noise levels in the dataset tested.

| | Models | Noise level (Standard deviation) 0 ($10^{-2}$) | 2 ($10^{-2}$) | 4 ($10^{-2}$) |
|---|---|---|---|---|
| SME | PI | 10.1±2.0 | 9.8±1.9 | 10.7±2.6 |
| | SHAP | 9.8±2.2 | 9.7±2.2 | 10.0±2.3 |
| | IG | 15.8±9.5 | 16.7±9.5 | 16.5±9.5 |
| MME | RATE (Kendall) | 8.8±3.2 | 8.8±3.2 | 9.4±3.6 |
| | RATE (Spearman) | 8.8±3.2 | 8.8±3.2 | 9.4±3.6 |
| | Median | 9.5±3.7 | 9.0±3.4 | 10.1±4.0 |
| | Mean | 8.8±3.2 | 8.7±3.2 | 9.4±3.6 |
| | Mode | 12.2±3.4 | 10.7±3.0 | 11.1±3.0 |
| | Box-whiskers | 9.1±3.3 | 9.1±3.3 | 9.5±3.6 |
| | Tau test | 8.9±3.3 | 8.8±3.2 | 9.5±3.6 |
| | Majority vote | **8.1±2.7** | **8.6±2.8** | **8.6±3.0** |

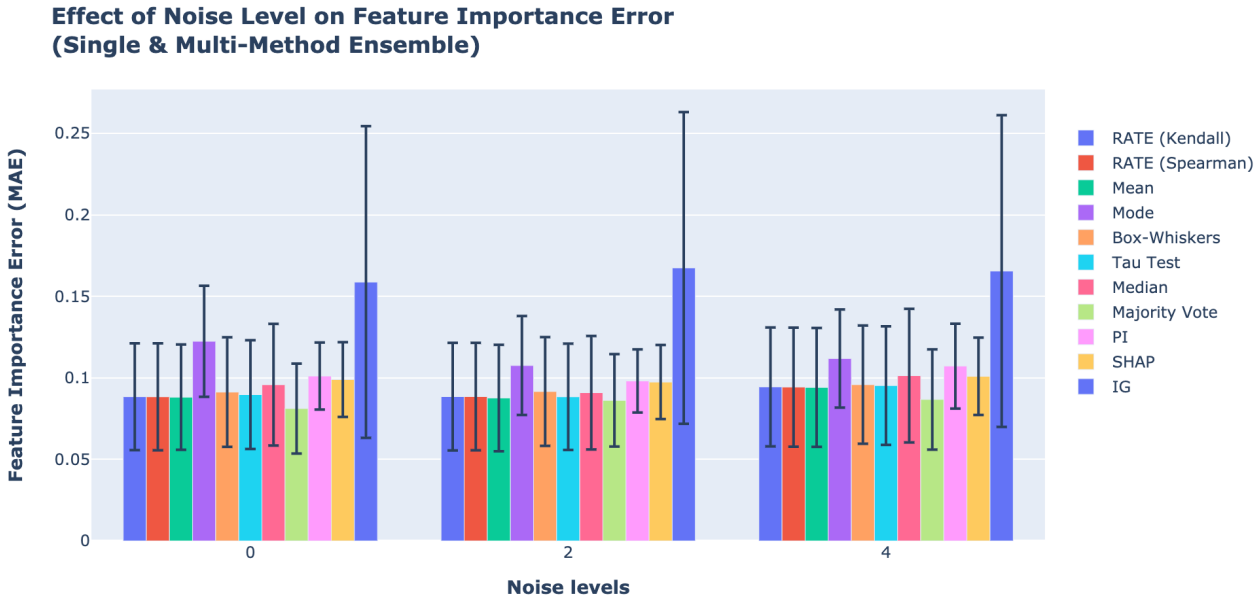



Figure 3.6: Effect of noise levels on SME and MME framework with decision fusion methods.

Figure 3.7 presents the FI MAE error for SME and MME frameworks adopting different fusion strategies averaged across 5 different feature informative levels in the data. Results on the effect of feature informative levels on ensemble FI using RMSE and $R^2$ are shown in Appendix A.1.5 and Appendix A.1.6. The best performing method is MME with majority vote followed by MME with mean and RATE. All decision fusion methods embedded into the MME framework except for mode outperform SME by more accurately quantifying the FI. The error bar (standard deviation) for the effect of features informative levels in Figure 3.7 has a smaller range than the effect of noise and the effect of number of features. The low standard deviation indicates that the feature informative levels explain most of the variances in the data. From Table 3.6 we observe that when 20% of the features are informative, the best FI methods are MME framework with Modified Thompson tau test and median for the dataset investigated, and they outperform the best SME method, SHAP, by 9.1%. For 40% features informative level, MME framework with RATE (Kendall and Spearman) and mean have the lowest error, with 6.9% improvement over SME with SHAP. For 60% features informative level, MME framework using RATE (Kendall and Spearman) have the best results, with a 4.1% improvement over SME with SHAP. Furthermore, MME framework with majority vote obtains the lowest error for both 80% and 100% features informative level. The best performing SME is using PI. MME with majority votes outperforms SME's best results by 25.4% and 23.0% for 80% and 100% features informative level, respectively.

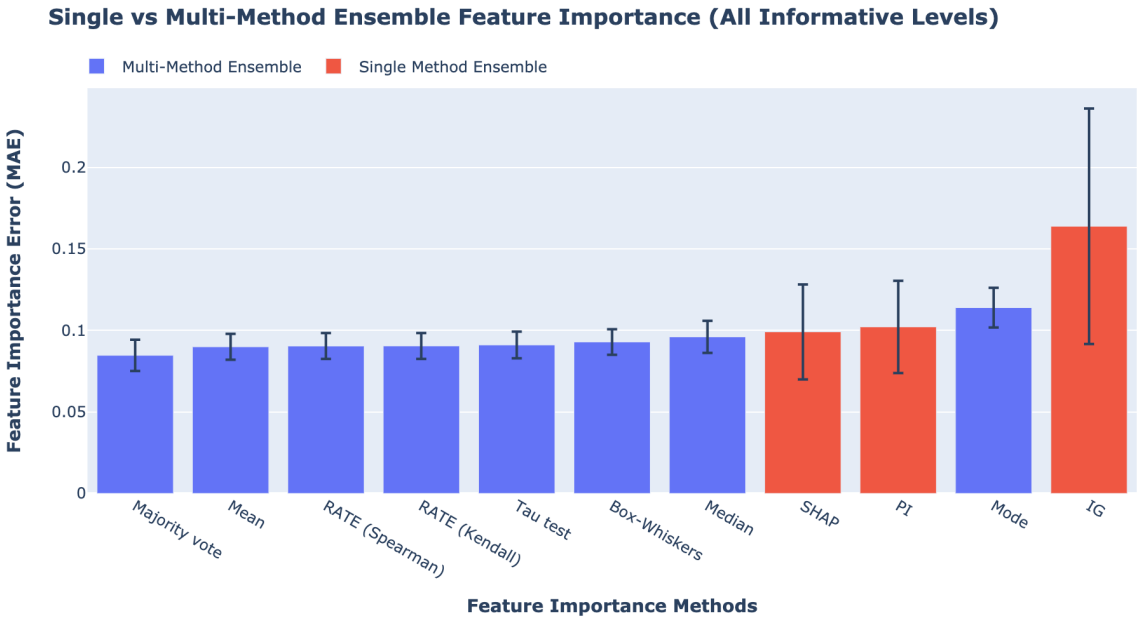



Figure 3.7: Effect of feature informative level on SME and MME framework with decision fusion methods.

Table 3.6: Summary of feature importance MAE between different SME and MME framework for different percentage of informative level.

| | | Features informative level (%) | | | | |
|---|---|---|---|---|---|---|
| | Models | 20 ($10^{-2}$) | 40 ($10^{-2}$) | 60 ($10^{-2}$) | 80 ($10^{-2}$) | 100 ($10^{-2}$) |
| SME | PI | 2.7±0.4 | 6.7±0.8 | 11.2±2.0 | 13.8±1.0 | 16.5±1.3 |
| | SHAP | 2.2±0.4 | 5.8±0.8 | 9.7±1.0 | 14.2±1.5 | 17.3±1.9 |
| | IG | 6.3±7.2 | 10.7±7.2 | 15.8±7.2 | 22.2±7.2 | 26.7±7.2 |
| MME | RATE (Kendall) | 2.1±0.4 | **5.4±1.1** | **9.3±1.5** | 12.3±1.6 | 15.9±3.0 |
| | RATE (Spearman) | 2.1±0.5 | **5.4±1.1** | **9.3±1.5** | 12.3±1.6 | 15.9±3.0 |
| | Median | **2.0±0.6** | 5.9±1.4 | 10.2±1.8 | 13.0±2.4 | 16.7±3.5 |
| | Mean | 2.1±0.5 | **5.4±1.1** | 9.4±1.5 | 12.3±1.6 | 15.7±3.0 |
| | Mode | 7.0±2.0 | 9.0±3.1 | 12.4±3.1 | 13.3±1.9 | 15.0±3.0 |
| | Box-whiskers | 2.1±0.6 | 5.6±1.1 | 9.5±1.4 | 12.9±1.7 | 16.1±2.9 |
| | Tau test | **2.0±0.6** | 5.6±1.2 | 9.5±1.5 | 12.4±1.7 | 15.7±3.0 |
| | Majority vote | 3.1±0.9 | 6.5±1.5 | 9.4±1.6 | **10.3±2.1** | **12.7±3.4** |

Figure 3.8 shows the relationship between the MAE of FI errors and the percentage of features informative level. As the percentage of feature informative level increases the FI errors increases. The higher number of contributing (non-zero) features to the output increases the difficulty of quantifying FI leading to higher errors.

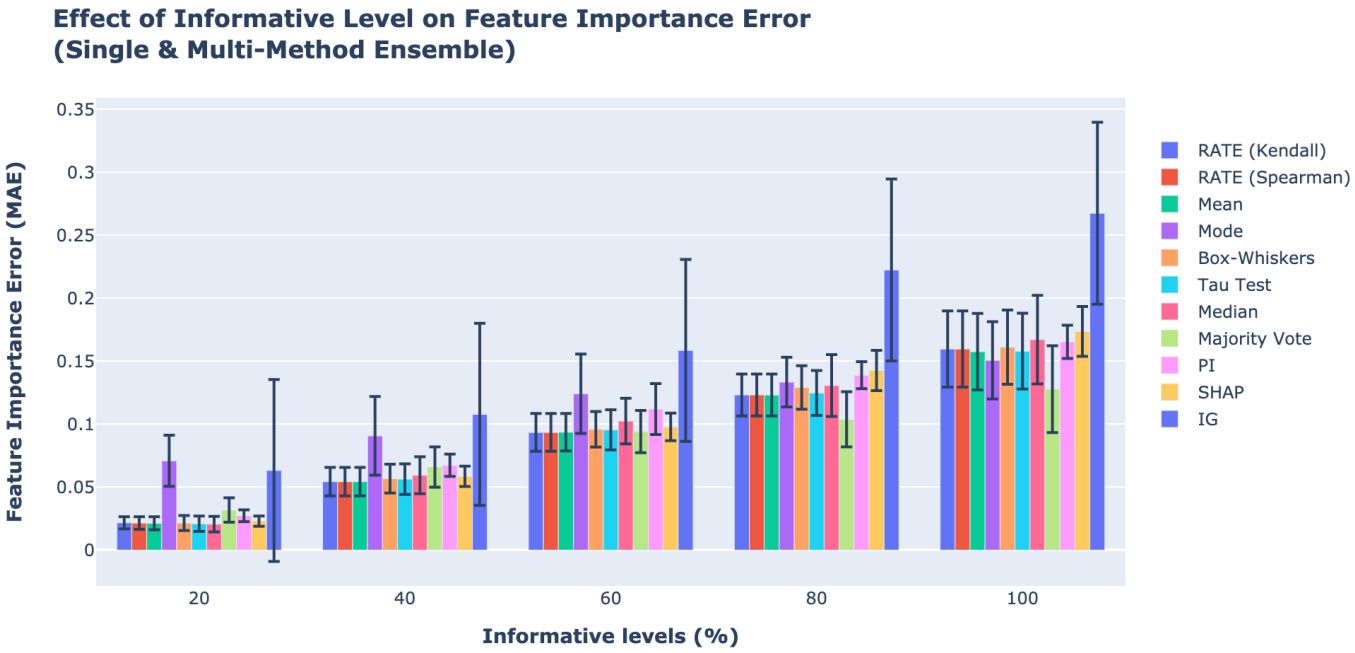



Figure 3.8: Effect of feature informative levels on SME and MME framework with decision fusion methods.

Figure 3.9 depicts the average of all SME and MME framework with fusion methods across 20, 60, and 100 features. Results on the effect of number of features on ensemble FI using RMSE and $R^2$ are shown in Appendix A.1.7 and Appendix A.1.8 of the supplementary material. Similar to the effect of noise level and the percentage of informative features on FI errors, MME framework with majority vote has the lowest error across all number of features. The error bar for the effect of feature number in Figure 3.9 has a smaller range compared to the FI errors of the effect of noise but larger than the effect of number of features. From Table 3.7 we observe that when there are 20 and 40 features the method with the lowest FI errors is MME framework with majority vote, and it outperforms the best SME method, SHAP,

by 8.2% and 26.2% respectively. For 100 features, MME framework using mode is the most accurate method, and it outperforms the best SME method, PI by 20.5%. For SME, PI has lower FI errors compared to SHAP as the number of informative features and features increases.

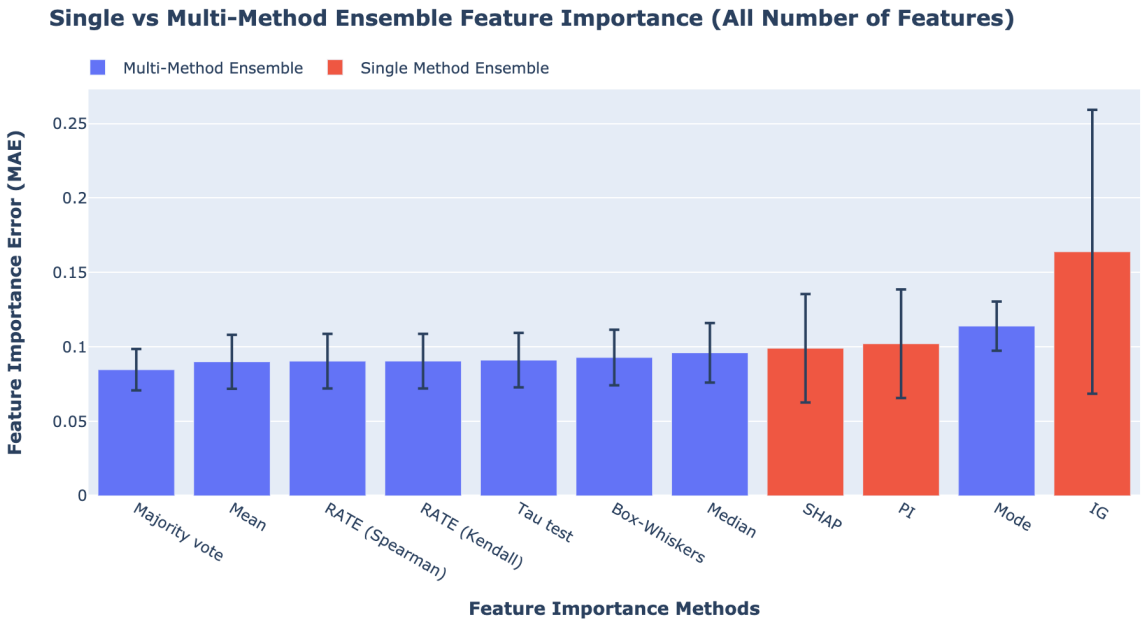



Figure 3.9: Effect of all number of features on SME and MME framework with decision fusion methods.

Table 3.7: Summary of feature importance MAE between different SME and MME framework for different number of features.

| | | Number of features | | |
|---|---|---|---|---|
| | Models | 20 ($10^{-2}$) | 60 ($10^{-2}$) | 100 ($10^{-2}$) |
| SME | PI | 7.5±1.8 | 10.8±2.0 | 12.2±2.4 |
| | SHAP | 6.1±1.5 | 10.3±2.2 | 13.1±2.4 |
| | IG | 15.8±9.5 | 16.1±9.5 | 17.1±9.5 |
| MME | RATE (Kendall) | 6.3±2.3 | 9.4±3.1 | 11.3±3.8 |
| | RATE (Spearman) | 6.37±2.3 | 9.4±3.1 | 11.3±3.8 |
| | Median | 6.0±2.3 | 10.6±3.8 | 12.0±3.9 |
| | Mean | 6.2±2.2 | 9.3±3.1 | 11.3±3.8 |
| | Mode | 14.8±3.6 | 9.6±2.3 | **9.7±2.4** |
| | Box-whiskers | 6.5±2.5 | 9.8±3.2 | 11.4±3.7 |
| | Tau test | 6.1±2.4 | 9.7±3.2 | 11.4±3.7 |
| | Majority vote | **5.6±1.9** | **7.6±1.5** | 12.1±3.3 |

Figure 3.10 shows the relationship between the MAE of FI and the number of non-orthogonal features as they increase. Higher numbers of features increase the difficulty of quantifying FI accurately.

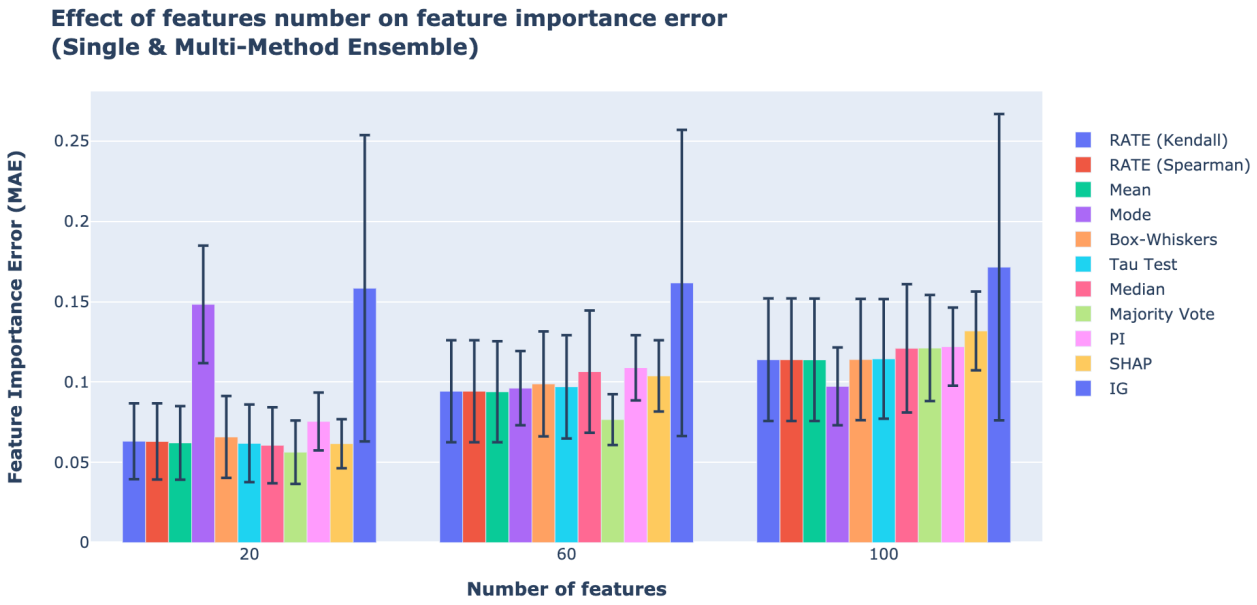



Figure 3.10: Effect of number of features on ensemble feature importance.

### Case Study 1 Results Discussion

Overall, results show that the MME framework outperforms SME for the case studies. In particular, the MME framework with majority vote as an ensemble strategy for the three different combinations of factors: (i) noise level, (ii) feature informative level and (iii) number of features has achieved the best results. The robustness of the framework compared to SMEs becomes even more evident as the number of features and the number of informative features increases. Results also reveal that the noise in data does not affect the final FI estimates, as there are no significant changes as the noise increases.

For the experiments, the hyperparameters of each ML models are kept constant as their case-based optimisation is likely to affect FI estimates for different data

characteristics. In the future, it is necessary to further investigate the interplay between parameter optimisation and FI. In addition, other factors such as covariate drift on the test dataset and imbalanced datasets should be included in the framework tests.

One advantage of using the model-agnostic MME framework is that it avoids the worst-case scenario for FI estimates due to the inherent advantage of ensembles being more robust against spread and dispersion of the predictions. Additionally, using only one set of explainability can potentially bias the FI estimates. Similarly, the FI estimates might be bias if only a single method is employed as it might not take into consideration the synergy between orthogonal features. By fusing FI estimates of multiple ML models and methods we can achieve a final FI estimates that is more robust. For safety-critical systems, it is important for the FI estimates to be robust to noise, biases, and anomalies for end users to accurately explain the results of ML models.

Conversely, this can also be a disadvantage as the best performing FI estimate might be moderated by worse methods. However, for real-world, save critical systems where the ground truth is unknown, it is important not to rely solely on a single method and the potential bias it might produce. In scenarios where all FI determined by different methods disagree or there is no consensus on FI, RATE or majority vote will be reduced to taking the average of all FI vectors. Under such circumstances, it is the best option for safety is to further investigate the reasons for disagreement between ML model or FI techniques. Furthermore, while majority vote achieves the best results for many different data characteristics, it is not always the case. Further

studies is necessary to decide which ensemble method is definitely better for specific data characteristics.

Despite MME's success over SME or non-ensemble approach, it still has several shortcomings. For example, an ensemble approach loses information as FI of individual instances or FI methods are compressed down to a single vector. The original FI from different methods before ensemble could potentially contain important information that can provide useful insights such as to which method could be an anomaly or how the output differs between FI methods. Furthermore, the output from the current FI approaches such as SME, MME or individual methods are not easily interpretable by experts. The numerical values of FI assigned to each features might be un-intuitive or even meaningless to experts of different domains.

More generally, for nonlinear response surfaces (the multidimensional surfaces encoded by trained ML models), the FI depends on the part of the response surface is calculated. The variation in feature gradients from one position to another results from the nonlinear behaviour of real-world systems. This variation means that the FI is not fixed for nonlinear models but is context-dependent. Sensible choices for the response surfaces to evaluate FI are typically local or global minima or maxima in the response variable, depending on what type of optimum is relevant.

Additionally, the characteristics of the learning process of ML models causes fluctuations in FI quantification. For example, RF combines the outputs of decision trees at the end of the process while gradient boosting machines [34] start combining at the start of the process. ML models employ several hyperparameters that need to be configured before training the model, and each configuration produces different

prediction and FI values. For example, random forests have the following hyperparameters: the number of trees, the maximum number of features to consider when splitting a node, and the minimum number of leaves to split an internal node.

Similarly, variation among different FI techniques also produces different results. For instance, PI is based on the decrease in model performance caused by the permutation of a feature, while SHAP is based on the contribution of features assessed by every possible combination. Those particularities in the calculation of importance by different techniques produce different values and interpretations of importance.

A solution to combine different FI results while capturing the uncertainties and nuances caused by the varied ML models and FI technique is necessary to provide a complete result.

## 3.6 Case Study 2: Main factors Affecting Creep Rates in Laser Powder Bed Fusion

In applications such as aerospace jet engines, components such as first stage turbine discs/blades operate under extreme temperatures and stresses. These components are critical and therefore cannot fail in service or enter service when faulty [119]. Turbine discs, for example, must be manufactured from materials with adequate mechanical properties, such as high fatigue and creep resistance, strength and mechanical integrity at elevated temperatures [120]. In particular, creep is one of the most significant causes of failure of such components as temperatures increase [121]. There is an increasing need to improve Additive Manufacturing (AM) for critical

application engineering components as it enables innovative designs, such as internal cooling channels, to easily be integrated and manufactured at no extra cost or time [122]. Laser Powder Bed Fusion (LPBF) is one of the main AM processes used for metal manufacturing and nickel-based superalloys, particularly alloy 718 as it have some of the best current LPBF printability [123]. However, the materials manufactured using AM perform well below their traditionally manufactured counterparts, particularly for creep and fatigue. Research has shown that this difference in performance is due to the complex relationships between AM process parameters affecting the material microstructures and their mechanical performance. Therefore, it is necessary to understand the impact of different AM build parameters on the mechanical performance of parts. For that, the MME framework is applied to understand the feature importance of different factors affecting the LPBF print. The data collected from this case study consists of 265 porosity images from printed alloy 718 with different AM build parameters. Using ML methods the creep rate is predicted using the porosity images. Subsequently, the material descriptors and build parameters affecting the creep rate are extracted using the MME framework. Detailed explanation of the data acquisition process of the original data and material descriptors generation from the porosity images, motivations, and objectives are described in Sanchez *et al.* [124].

### 3.6.1 Data Preparation for Machine Learning Models

Data preparation is conducted to ensure that the data is suitable to be trained by the ML models. First, the 265 samples (porosity images) are cropped approximately in

half to produce 512 samples as part of the data augmentation. Data augmentation is used to increase the amount of data by modifying existing data. Data augmentation is also emploted as a regulariser for ML models to prevent overfitting [125]. From the images, material descriptors are extracted using the *Python scikit-image* [126] library. The *scikit-image* library measures various properties for each connected region labelled in the binary images, more specifically the *regionprops* function under the *measure* modules. The new material descriptors generated contain information regarding the geometrical shape and mathematical properties in continuous values. The categorical features are transformed into numerical data using the label encoding method [127], as shown in Figure 3.11.

| Scan Strategy | Number of Lasers | Build Orientation |
|---|---|---|
| Meander | Single | Vertical |
| Stripe | Multiple | Horizontal |
| Meander | Single | 45 degree |

| Scan Strategy | Number of Lasers | Build Orientation |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 0 | 0 | 2 |

Figure 3.11: Transforming categorical data to numerical values using label encoding method.

Subsequently, the continuous material descriptors are scaled to [0, 1] range. The material descriptors are scaled to ensure equal weighting is given to all material descriptors to prevent material descriptor bias in training. Next, the dataset is split into training and testing data instances (each data instance consists of material descriptors from one porosity image).

The purpose of this data is to investigate the predictability of creep rate when all test cases are included in both training and test data. Leave-One-Case-Out (LOCO) is set up to investigate the ML models creep rate prediction accuracy when it is

trained in the absence of one unseen sample. LOCO is the direct application of Leave-One-Out cross validation where a subset of data is left out to be tested while the remaining is used to train the model. LOCO is designed to investigate if ML can accurately predict the creep rate of unseen test cases. For example, the ML models are first trained on data with the samples of Vertical Single laser Meander (VSM), Vertical Single laser Stripe (VS), Vertical Multi Laser (VM), 45 degree Single Laser (45S), 45 degree Multi Laser (45M), and Horizontal Single Laser (HS) and then tested on Horizontal Multi Laser (HM) data. The cycle continues for all test cases, as shown in Figure 3.12. The ML models are first trained on the first nine subsets of data and tested on the last subset of data. In the subsequent iteration, ML models are trained on the first eight and the last subset of data and tested on the ninth subset data. The results obtained from each iteration are collected, and the final result is then averaged across all test case iterations.

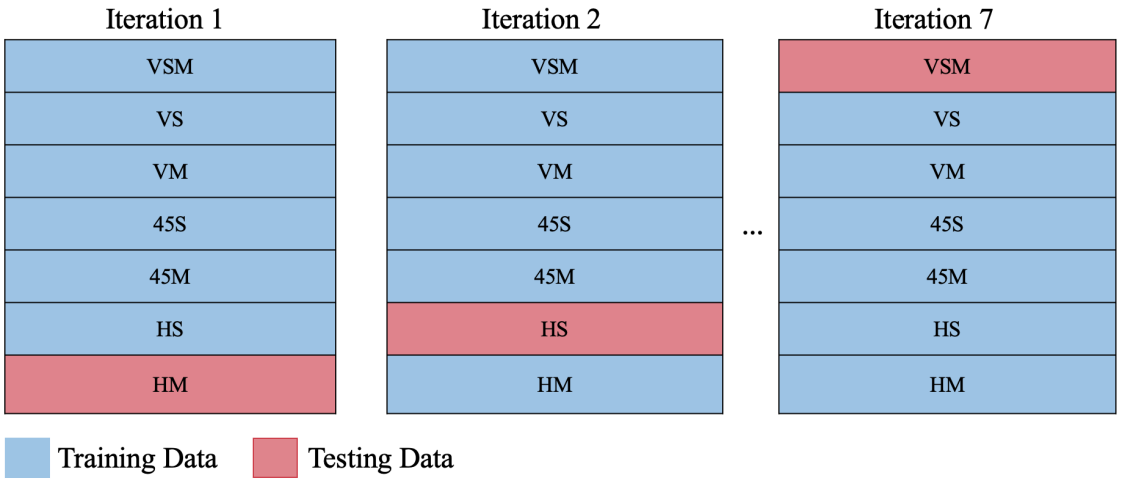



Figure 3.12: Evaluation strategy of ML models when one test case is excluded from the training data. The excluded test case is used as the testing data. Testing is repeated for each test case.

### 3.6.2 Machine Learning Methods

The ML methods selected for the experiment are RF, GBT, DNN, SVR, Ridge Regressor, and LASSO Regressor. Regularised linear models such as Ridge and LASSO Regressors are chosen as a baseline to verify whether a more complex method is required to learn the pattern in data. RF and GBT are selected as both models have shown great results for structured and tabular data [128]. Furthermore, DNN [129, 130, 131] and SVR [132, 133] are also investigated as several AM papers have shown great result using these methods. All ML models are implemented using *scikit-learn* [134] except for DNN, which are implemented using Tensorflow [135].

### 3.6.3 Evaluation Metrics

The metrics used to evaluate the accuracy of predicted creep rate were Median Absolute Deviation (MAD), Coefficient of Determination ($R^2$), MAE, RMSE, and Mean Absolute Percentage Error (% Error). Multiple evaluation metrics allow for better understanding of the error produced by ML models [136]. MAE and RMSE both measure the average magnitude of the error. A larger error is penalised more by RMSE, while MAE provides a linear penalty to the magnitude of the error. MAD is a robust measure to outlier in error. $R^2$ measures the goodness of fit of predicted output to the actual result. Finally, % Error describes the deviation of error from the actual result in percentage form for more natural understanding.

### 3.6.4 Results

Figure 3.13 and Table 3.8 shows the results for our experiments using LOCO. The best results for each left out conditions are in bold text in the Table. RF performs the best in predicting the creep rate for VS test cases when they are left out of the training data. GBT achieves the best creep rate prediction for 45M and VM test cases while SVR predicted 45S, HM, HS, and VSM creep rate with the lowest error. Overall, the creep rate prediction for 45M, 45S, HS, and VS have the lowest error at less than 20%, as indicated by the % Error evaluation metric. The predicted creep rate for HS achieves the lowest % Error at 1.40%. Predicted VM and HM creep rate % Error are higher compared to 45M, 45S, HS, and VS at 48.14% and 35.80% respectively. The highest predicted error observed is for the VSM test case. The prediction for VSM creep rate is considered non-predictive by RF, GBT and DNN, as the % Error error were greater than 400% for each model but SVR were able to narrow the error down to 60.68%. Furthermore, the result of LASSO Regressor had low error for VS creep rate prediction. However, further investigation showed that the LASSO model had is predicting the average value of all creep rate which coincidentally is very close to the VS creep rate resulting in low error but the model itself did not learn the data pattern. Between all the models tested, DNN had the highest uncertainty in its prediction while RF GBT, SVR, Ridge Regressor, and LASSO Regressor prediction had low uncertainty throughout ten repeated experiments. For those reasons, only RF, GBT, and SVR were included in the framework to determine the most important material descriptors that affected the predicted creep rate as these three methods had the best prediction performance.

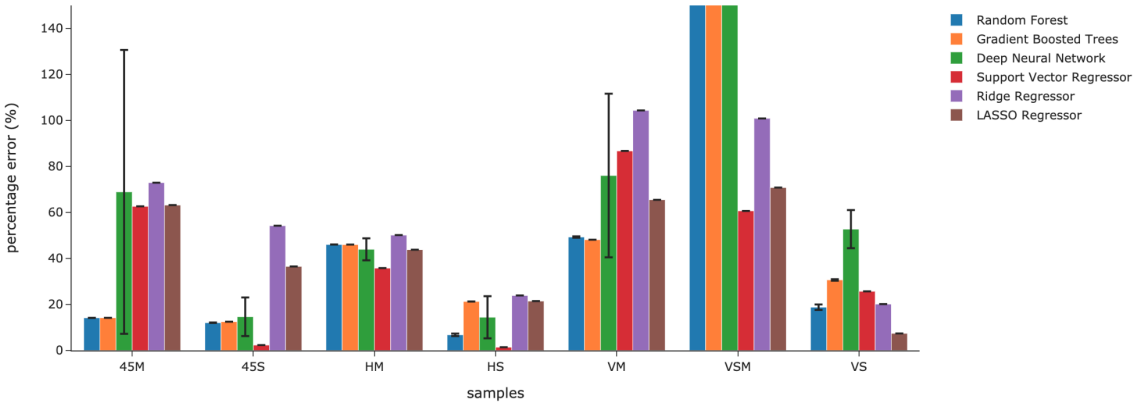



Figure 3.13: Percentage error of creep rate predictions for Leave-One-Condition-Out experiment using Random Forest, Gradient Boosted Trees, Deep Neural Network, Support Vector Regressor, Ridge Regressor, and LASSO Regressor. The Y-axis shown here is limited to the range of 0-150% to provide a better view as the percentage error for VSM predicted by Random Forest, Gradient Boosted Trees, and Deep Neural Network are greater than 400%.

Table 3.8: Creep rate prediction results for Leave-One-Condition-Out experiment.

| Models | Evaluation Metrics | 45M | 45S | HM | HS | VM | VSM | VS |
|---|---|---|---|---|---|---|---|---|
| Random Forest | MAD | 4.31 ± 0.01 | 4.24 ± 0.02 | 26.09 ± 0.00 | 1.55 ± 0.41 | 13.42 ± 0.08 | 100.10 ± 0.00 | **7.50 ± 0.76** |
| | MAE | 4.32 ± 0.01 | 4.19 ± 0.03 | 26.07 ± 0.02 | 3.33 ± 0.27 | 13.46 ± 0.07 | 100.10 ± 0.00 | **7.59 ± 0.49** |
| | % Error | 14.17 ± 0.04 | 12.05 ± 0.11 | 46.06 ± 0.04 | 6.78 ± 0.55 | 49.32 ± 0.28 | 418.82 ± 0.00 | **18.80 ± 1.21** |
| | RMSE | 4.32 ± 0.01 | 4.19 ± 0.03 | 26.07 ± 0.02 | 4.12 ± 0.27 | 13.46 ± 0.07 | 100.10 ± 0.00 | **8.75 ± 0.31** |
| Gradient Boosted Trees | MAD | **4.31 ± 0.00** | 4.28 ± 0.00 | 26.08 ± 0.00 | 7.41 ± 0.00 | **13.17 ± 0.00** | 100.09 ± 0.00 | 13.17 ± 0.07 |
| | MAE | **4.31 ± 0.00** | 4.35 ± 0.01 | 26.05 ± 0.01 | 10.47 ± 0.02 | **13.14 ± 0.00** | 100.09 ± 0.00 | 12.39 ± 0.13 |
| | % Error | **14.19 ± 0.02** | 12.50 ± 0.02 | 46.02 ± 0.01 | 21.29 ± 0.04 | **48.14 ± 0.01** | 418.82 ± 0.00 | 30.67 ± 0.32 |
| | RMSE | **4.32 ± 0.01** | 4.35 ± 0.01 | 26.05 ± 0.01 | 10.99 ± 0.02 | **13.14 ± 0.00** | 100.09 ± 0.00 | 12.52 ± 0.11 |
| Deep Neural Network | MAD | 20.96 ± 18.92 | 4.69 ± 3.16 | 25.39 ± 2.46 | 6.73 ± 5.10 | 20.07 ± 10.68 | 99.86 ± 0.18 | 14.62 ± 5.73 |
| | MAE | 21.02 ± 18.92 | 5.10 ± 3.16 | 24.88 ± 2.46 | 7.10 ± 5.10 | 20.76 ± 10.68 | 98.55 ± 0.18 | 21.30 ± 5.73 |
| | % Error | 68.94 ± 61.69 | 14.65 ± 8.40 | 43.96 ± 4.80 | 14.43 ± 9.16 | 76.06 ± 35.55 | 412.38 ± 2.66 | 52.73 ± 8.30 |
| | RMSE | 21.40 ± 18.5 | 5.69 ± 2.70 | 35.10 ± 2.79 | 7.97 ± 4.31 | 21.97 ± 10.37 | 98.62 ± 0.59 | 27.70 ± 4.91 |
| SVR | MAD | 19.33 ± 0.00 | **0.75 ± 0.00** | **20.12 ± 0.00** | **0.51 ± 0.00** | 23.68 ± 0.00 | **14.64 ± 0.00** | 10.26 ± 0.00 |
| | MAE | 19.11 ± 0.00 | **0.81 ± 0.00** | **20.26 ± 0.00** | **0.69 ± 0.00** | 23.68 ± 0.00 | **14.50 ± 0.00** | 10.37 ± 0.00 |
| | % Error | 62.64 ± 0.00 | **2.34 ± 0.00** | **35.80 ± 0.00** | **1.40 ± 0.00** | 86.74 ± 0.00 | **60.68 ± 0.00** | 25.69 ± 0.00 |
| | RMSE | 19.12 ± 0.00 | **0.93 ± 0.00** | **20.26 ± 0.00** | **0.91 ± 0.00** | 23.68 ± 0.00 | **14.51 ± 0.00** | 10.41 ± 0.00 |
| Ridge Regressor | MAD | 22.29 ± 0.00 | 19.23 ± 0.00 | 28.20 ± 0.00 | 12.06 ± 0.00 | 28.44 ± 0.00 | 22.67 ± 0.00 | 4.22 ± 0.00 |
| | MAE | 22.25 ± 0.00 | 18.87 ± 0.00 | 28.39 ± 0.00 | 11.75 ± 0.00 | 28.49 ± 0.00 | 24.11 ± 0.00 | 8.15 ± 0.00 |
| | % Error | 72.96 ± 0.00 | 54.22 ± 0.00 | 50.17 ± 0.00 | 23.89 ± 0.00 | 104.36 ± 0.00 | 100.89 ± 0.00 | 20.17 ± 0.00 |
| | RMSE | 22.26 ± 0.00 | 18.95 ± 0.00 | 28.40 ± 0.00 | 11.89 ± 0.00 | 28.49 ± 0.00 | 24.90 ± 0.00 | 11.51 ± 0.00 |
| LASSO Regressor | MAD | 19.27± 0.00 | 12.71 ± 0.00 | 24.80 ± 0.00 | 10.57 ± 0.00 | 17.88 ± 0.00 | 16.92 ± 0.00 | 2.99 ± 0.00 |
| | MAE | 19.27 ± 0.00 | 12.71 ± 0.00 | 24.80 ± 0.00 | 10.57 ± 0.00 | 17.88 ± 0.00 | 16.92 ± 0.00 | 2.99 ± 0.00 |
| | % Error | 63.20 ± 0.00 | 36.52 ± 0.00 | 43.82 ± 0.00 | 21.48 ± 0.00 | 65.51 ± 0.00 | 70.82 ± 0.00 | 7.41 ± 0.00 |
| | RMSE | 19.27 ± 0.00 | 12.71 ± 0.00 | 24.80 ± 0.00 | 10.57 ± 0.00 | 17.88 ± 0.00 | 16.92 ± 0.00 | 2.99 ± 0.00 |

*Note: All numbers are in* 1e−4 *except % Error.*

Subsequently, Figure 3.14 shows the ensemble material descriptor importance using the MME framework and majority vote as the decision fusion method for the top three most performant ML models i.e. RF, GBT, and SVM for the LOCO experiment. The four most important material descriptors used by the three ML models to predict creep rates were the density, number of pores, build orientation, and scan strategy in decreasing order. The importance of material descriptors for density, number of pores, build orientation, and scan strategy were 23.0%, 21.9%, 17.7%, and 13.8% respectively. The four most important material descriptors accounted for 76.4% of importance. The remaining fifteen material descriptors were considered less important and had low contributing factor as they only accounted for 23.6% of material descriptor importance.

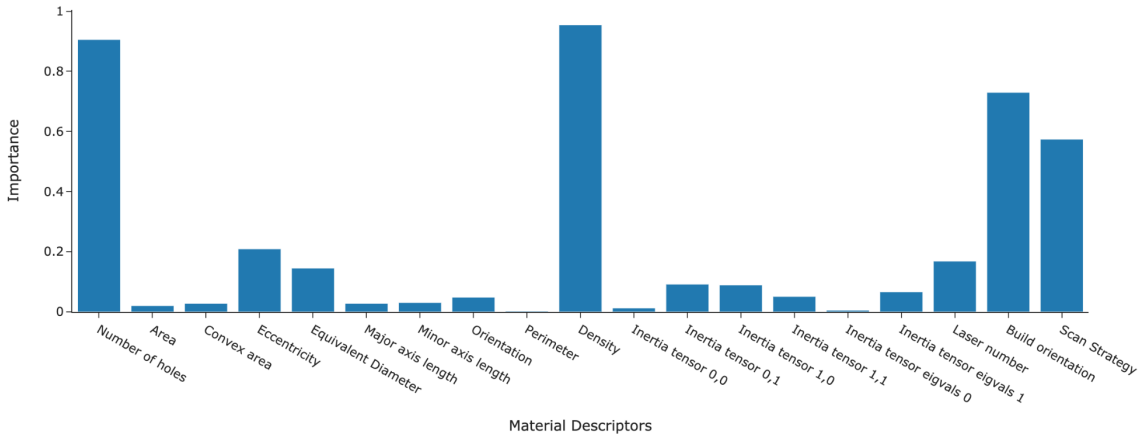



Figure 3.14: Ensemble material descriptor importance for Leave-One-Case-Out experiment.

### 3.6.5 Discussions

In the LOCO experiment, DNN experienced difficulty in generalising the model for all test cases. Similarly, Ridge and LASSO Regressors failed to map the relationship between material descriptors and creep rate. RF, GBT, and SVR were able learn the data pattern for 45M, 45S, HS, and VS when they were left out of the training set. When test cases share similar print conditions but have different creep rate e.g. HS and HM, the ML models attempts to generalise both test cases together which leads to high errors. Additionally, some combinations of test cases in the training data might provide more vital information to assist ML models to generalise better.

The ensemble material descriptor importance obtained from the ML models using MME was able to accurately identify the most important material descriptors affecting creep rate. Using ML models and interpretable methods allowed information such as important material descriptors that would otherwise have been difficult to obtain using traditional methods or more extensive experimentation. Although FEA may result in slightly more accurate creep rate prediction, it is unable to give an explanation of which factors causes these fluctuations in the manufacturing process. Additionally, the adaptation of FEA models for AM specific characteristics, such as build parameters and porous microstructure, has yet to be addressed.

Therefore, ML is not discounted as a powerful tool for AM. Multiple build parameters can be included in ML models, unlike FEA. These inputs can be used for density prediction, mechanical property prediction and more. One downside of ML is that most models require a lot of data. The more data available to train the model, the more accurate that model will be. Indeed, predicting the creep rate using

LOCO was obtained from a population size of 512 images whereas most ML models have a population size in the hundreds of thousands, resulting in extremely accurate predictions. By inputting more experimental data, the models' accuracy should improve [137]. Thus, the findings here are limited to the small available data as it is difficult to generate large creep dataset, and should be further validated in the future, as more data is made available. Additionally, it is difficult to empirically select which ML method or set of ML methods in the case of ensemble will perform well in its prediction task before starting the experiment. Therefore a lot of trial-and-error or reliance on experience is necessary to select the right set of ML methods. In addition to the difficulty in ML method selection, the process of understanding the output of MME can be challenging for non-experts. While the MME are able to output reasonably accurate material descriptor importance it is not obvious to the domain experts how did MME come to such conclusion about the importance, what does the value of importance mean, and how each ML methods contribute the final outcome. The lack of transparency into the result of MME interpretation method add another layer of unintended obfuscation to the domain experts who requires more clarity into the material descriptor importance. Finally, another shortcoming of MME framework is that while it extracts knowledge more accurately and robust than single interpretation technique, it is not able to incorporate prior domain knowledge that could improves it interpretation accuracy.

## 3.7 Summary

In summary, this chapter introduced the MME framework — a crisp FI fusion of multiple ML models and FI methods with the aim of reducing variance in FI estimates. The main objective is to develop and test an ensemble FI framework for decision fusion that is more accurate in FI estimates compared than the existing FI method. A survey on related work in ensemble FI approach is also discussed along with a background on PI, SHAP, and IG. Subsequently, the MME framework is introduced and explained. The MME is then tested on two case studies with the first case study being synthetic dataset with multiple different conditions to compare the performance of traditional FI approaches and MME framework. The case study shows that MME is able to produce more accurate FI quantification compared to traditional FI approaches. For the second case study we employ MME framework to determine the main material descriptors and build parameters affecting the creep rate of printed alloy 718. The results show that the MME framework is able accurately explain the prediction produced by ML models. While MME is able to produce more accurate FI values compared to traditional approaches, it lacks the ability to capture uncertainties, incorporate prior domain knowledge, and interpret FI output in linguistic terms. In the next chapter, a Fuzzy extension to the MME framework is proposed to solve the shortcomings of the current approaches described above.

# Chapter 4

# The Fuzzy Multi-Method Feature Importance Ensemble Framework

An initial attempt to address the uncertainty in FI determination is the introduction of our MME framework in the previous chapter, where multiple FI approaches are applied on multiple ML models, and the crisp important values are combined to produce the final importance for each feature. A crisp value is the exact expression of a measurement. The overall results of MME are more robust and accurate (15% less feature importance error) for synthetic data sets when compared to using single ML and FI. However, MME does not provide a comprehensive exploration of feature causality as MME itself produces some uncertainties that is not captured within the framework. The crisp-based approach also loses information as the uncertainties are not taken into account during the decision fusion step. These uncertainties in identifying the contribution of features to ML outputs are due to:

1. For nonlinear response surfaces (the multidimensional surfaces encoded by trained ML models), the FI depends on which part of the response surface is interrogated. The variation in feature gradients from one position to another is a result of the nonlinear behaviour of real-world systems. This variation means that for nonlinear models, the FI is not fixed, but is context dependent. Sensible choices for the part of the response surfaces at which to evaluate FI are typical local or global minima or maxima in the response variable, depending on what type of optimum is relevant.

2. The variance due to the characteristics of the learning process of ML models. For example, RF combine the outputs of decision trees at the end of the training process while GBT combine them at the start of the process. Although random forest and gradient boosting machines are tree-based methods that use impurity-based methods to calculate feature importance, the final feature importances for two ML methods will differ due to their variations in training algorithms.

3. Differences in how FI is calculated and interpreted by FI techniques. For instance, certain approaches, such as PI investigate how each feature affects the model response individually. Here, PI shuffles the instance values for a particular feature, while maintaining the original values for the remaining features. The feature with shuffled instances is considered important if the model's performance decreases. Other approaches, however, investigate the importance of the feature both individually and in synergy with other subsets of features. For example, SHAP calculates the contribution of features for every possible

combination of the feature set investigated.

4. FI coefficients in general being calculated as the average (or weighted average) of the importance of a feature within a data sample. Information about the context (or data subspace) in which a feature has higher or lower importance during training is lost.

Furthermore, the use of crisp fusion in MME can also result in significant loss of information as data subspaces and gradients are not adequately explored because of the way training/test sampling is conducted. Crisp ensembles like majority voting discard extreme values of FI which, for safety critical systems, should not be disregarded. Additionally, SME and MME have limited interpretability; the methods and outputs are difficult for domain experts to understand. For example, what does a final FI coefficient of 0.76 mean (is it a good high value, or a low value)? The value obtained also depends on the FI method adopted and those that are not model agnostic also depend on the specific ML model(s) employed. Finally, to the best of our knowledge there is limited literature that explores data sampling for FI. In this chapter, it is shown how the limitations can be alleviated using a fuzzy logic approach for representation of importance. This accounts for multiple levels of uncertainty, whilst simultaneously simplifying interpretation of results. Fuzzy logic defines data/context-dependent intervals of importance that make it easier to identify data regions where specified features have high or low importance for decision making. This level of interpretation granularity will accelerate data-driven intelligent research and decisions by identifying critical data instances that enhance prognosis and health management, manufacturing, medicine and design of new materials.

To utilise fuzzy logic approach for representation of importance it is combined with MME to create Fuzzy Ensemble Feature Importance (FEFI), a method that captures and models the variance amongst different ML methods employed for FI calculation and the data space representation is proposed in this chapter. The objective of FEFI is to capture the uncertainty within ensemble feature importance to accurately explain ML models' output while presenting the feature importance in linguistic term for straightforward explanation and capturing the uncertainties that arises from the feature importance explanation. FEFI combines crisp importance values, and rules expressed in simple terms that explains a feature's importance. First, a data-driven approach to generate Membership Functions (MF) [138] is employed for the FI coefficients produced by re-sampling the dataset (e.g., by cross-validation), employing an ensemble of ML and FI methods. This allows the variance among the ML models, FI techniques, and data to be modelled as MFs [139] with three fuzzy sets; 'low', 'moderate' and 'high' representing the categories of FI. The Wang-Mendel [140] is employed to learn the rules that map the feature importance values generated by the different ML methods to the final importance. The Mamdani inference [141] is used to combine the FI terms generated by the MFs, using rules to produce a final description of a feature's importance in terms of simple terms for easy interpretation and decision-making.

For the remainder of this chapter, a background on Fuzzy Logic and FEFI is first explained in Section 4.1 and Section 4.2. Subsequently, the generation and pre-processing of the synthetic data along with a real-world case study of feature importance quantification of main factors affecting the secondary creep rate predic-

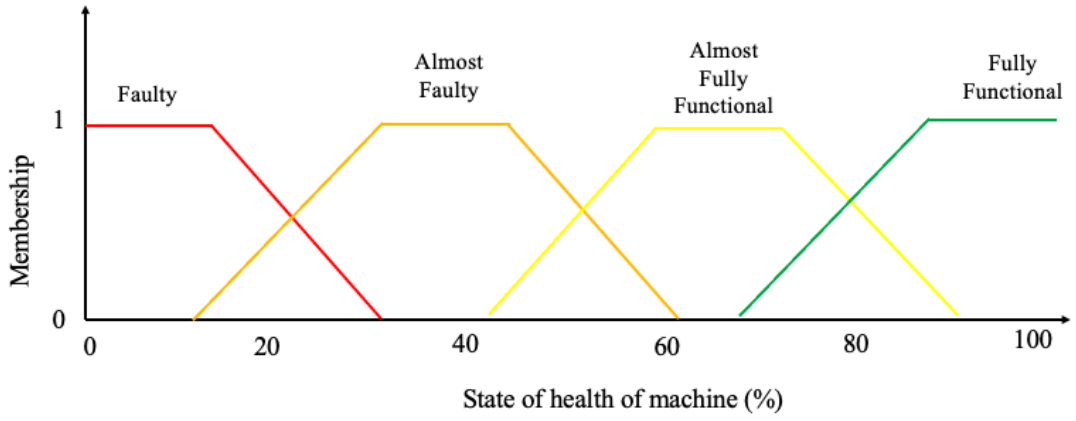
tion of Laser Powder Bed Fusion Material to test the performance of FEFI relative to MME in Section 4.4 and Section 4.5. Finally, the results and future work are discussed.

## 4.1 Background

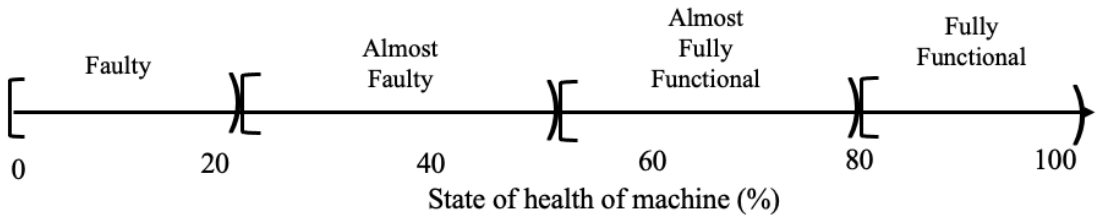
### 4.1.1 Fuzzy Sets

In 1965 Lotfi A. Zadeh published the seminal paper on Fuzzy Sets [139] that extends the classical set theory. In classical set theory, an element is either in a set, or it is not, adhering to the principle of bivalence in formal logic where the elements are discriminated between members and nonmembers using the degree of $[0, 1]$. This is also known as the crisp set in Fuzzy system. Fuzzy sets extend upon crisp sets by enabling elements to fall within a specific interval range, also indicating their membership values. Membership value is analogous to probabilities. The membership degree of an element is defined by a function called MF. The difference between a crisp set and fuzzy set is best illustrated on a graph. In Figure 4.1, the graphs (a) and (b) indicates the health of a machine ranging from faulty (0%) to fully functional (100%) on the x-axis. In Figure 4.1(a) fuzzy sets are also representing four linguistic concepts: faulty, almost faulty, almost fully functional, and fully functional. In the real world, these concepts are generally provided by experts on this machine or obtained from extensive data collection. Each concept is defined by the MFs and it ranges from zero to one. Within the fuzzy sets, the state of health of the machine is considered a fuzzy variable. A fuzzy variable is able to capture the uncertainties

compared to a crisp variable. For example, in Figure 4.1a, at around 85% state of health the machine is given the degree of 0.3 almost fully functional and 0.7 fully functional. Whereas for the crisp variable, 85% state of health is considered fully functional. The uncertainties caused by unavoidable measurements errors can be captured by using fuzzy variables.




(a)




(b)

Figure 4.1: State of health of a machine represented with (a) a fuzzy sets and (b) a crisp sets

### 4.1.2 Fuzzy Rule Generation

Each collection of fuzzy sets can be thought of as a rule, for example, in the case of the state of health of a machine, a specific range of health is mapped to one or two of the four linguistic concepts. Mathematically it can be represented as follows:

$$\text{Rule } R_i : \text{IF } x_1 \text{ is } A_{j1} \wedge ... \wedge \text{IF } x_m \text{ is } A_{jm} \text{ THEN } C_j \text{ with } w_j \tag{4.1}$$

where $j$ represents the number of rule, $A_j$ is the fuzzy set, $C_j$ is the linguistic concepts/class and $w_j$ is the rule associated weight. In the state of machine health example, the rule is defined by an expert. However, in the absence of experts, rules have to be automatically generated. One way is to generate the rules from data using the Wang-Mendel method [142] using input-output data pairs, similar to labelled data in the context of supervised learning. The Wang-Mendel method is comprised of four steps:

**1. Splitting the Input and Output Spaces into Fuzzy Regions**

Assuming the input data has two features $x_1$, $x_2$ and one output $y$. Their domain intervals are denoted by $[x_1^-, x_1^+]$, $[x_2^-, x_2^+]$, and $[y^-, y^+]$. The intervals for each input variable is the upper and lower limit on the $x - axis$ and it is divided into $2N + 1$ regions spaces between upper and lower limit. Each divided region now corresponds to a triangular shape MF, as shown in Figure 4.2, as opposed to trapezoidal shape MF previously shown in Figure 4.1. The overlapping area between the shapes, also known as fuzzy regions, captures the uncertainty of a variable. Since the number of

regions are divided into odd numbers, there is always a center region denoted by CE. The regions to the left side of CE are denoted by $S1, ...SN$. Similarly, the regions to the right of CE are denoted by $B1, ..., BN$ where $S$ and $B$ means small and big, respectively. Membership values range from $[0, 1]$. In the original paper, the authors proposed triangular MFs. However, other shapes of MFs are acceptable.

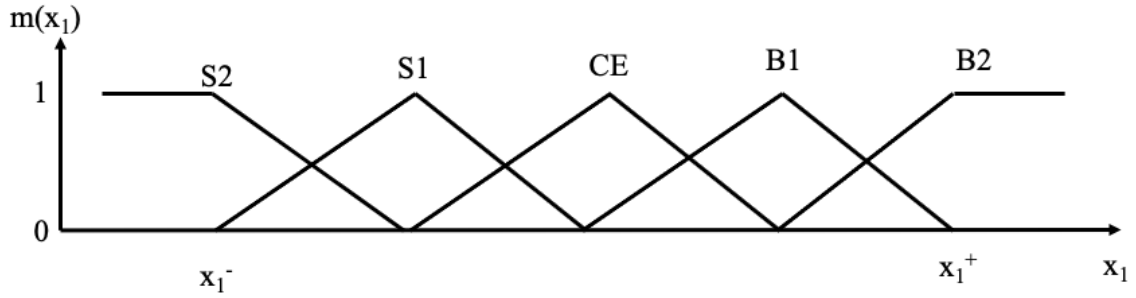



Figure 4.2: Division of fuzzy regions and corresponding membership function for a variable.

**2. Generating Fuzzy Rules From Data**

First convert each data into its membership degree given the MF in step 1. For example, the first instance of data input-output pair, $x_1^{(1)}$ has a degree of 0.4 in $S1$, $x_2^{(1)}$ has a degree of 1.0 in $CE$, and $y^{(1)}$ has a degree of 0.8 in $B2$. Then the rule can be represented as follows:

$$\text{Rule 1 : IF } x_1^{(1)} \text{ is (0.4in S1)} \wedge \text{IF } x_2^{(1)} \text{ is (1.0in CE) THEN } y^{(1)} \text{ is (0.8in B2)} \quad (4.2)$$

Rule 1 can be summarised as IF $x_1$ is $S1$ and $x_2$ is $CE$ THEN $y$ is $B2$. Each data

pair generates a rule. If there are a thousand data pair instances then a thousand rules are generated.

### 3. Eliminating Conflicting Rules

As the number of rules generated increases the probability of conflicting rules also increases. For example, Rule 1 is IF $x_1$ is $S1$ and $x_2$ is $CE$ THEN $y$ is $B2$ and Rule 2 is IF $x_1$ is $S1$ and $x_2$ is $CE$ THEN $y$ is $CE$, then Rule 1 and 2 are conflicting because the same IFs statement produces different THEN output. To resolve conflicting rules, each rule is assigned a membership degree. Only the rule with the highest degree within the conflicting group is accepted. The membership degree for rules is calculated using the membership degree of each variable. For example, Rule 1 from Equation 4.2 has a degree of $0.4 * 1.0 * 0.8 = 0.32$. Thus, Rule 2 is rejected if it has a lower degree compared to Rule 1.

### 4. Mapping The Fuzzy Rules

To obtain a new output $y$ using new instances the input $x_1, x_2$ are passed through the rules to produced a membership degree. However, the membership degree of $m(y)$ is not mapped to the real world values, therefore, a defuzzification step is required. One way to defuzzified fuzzy variable is to use the centroid defuzzification formula as follows:

$$y = \frac{\sum_{i=1}^{K} m_{O^i}^i, \bar{y}^i}{\sum_{i=1}^{K} m_{O^i}^i} \tag{4.3}$$

where $m_{O^i}^i$ is the product of all the input membership degree (in this case $m(x_1)$

and $m(x_2)$). The $O^i$ denotes the output region, e.g., CE, B1, S1, etc of Rule $i$. $\bar{y}^i$ represents the center value of region $O^i$ and $K$ is the number of fuzzy rules.

By following Step 1-4, one can produce a fuzzy inference systems that maps the input space to an output space while capturing uncertainties along the way. Similar to neural networks, Wang-Mendel method is also a universal approximator and can be extended to support a general n-input and n-output system. In the case of this chapter, Wang-Mendel method is employed to generate the rules to combine the MFs generated by the importance coefficients to produce final feature importance with uncertainties from ML models, FI values, and data captured and reflected.

## 4.2 Fuzzy Ensemble Feature Importance

This section describes the FEFI methodology using multiple ML models and FI techniques. First, the data goes through pre-processing stage. Second, multiple ML models are applied to the pre-processed data to capture the variance in the entire dataset. Cross validation is used to tune hyperparameters and to understand the characteristics of the ML models and feature gradients. Third, multiple model-agnostic FI methods are used to analyse the trained ML models to quantify FI. Finally, these importance values are then used to train a Fuzzy Inference System (FIS) as shown in Figure 4.3. The stages of the methodology are further described below.

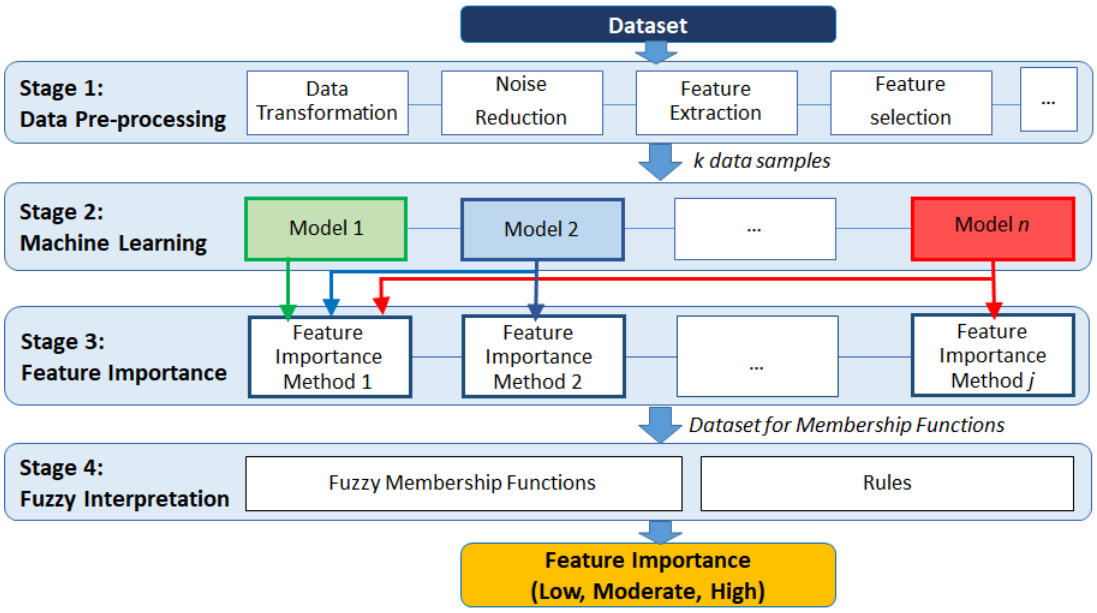



Figure 4.3: The four stages of the fuzzy ensemble feature importance method. The first stage pre-processes the data and the second stage partitions the data into $k$ subsets for training and evaluation. The third step employs multiple FI techniques to calculate feature importance from the each trained ML model. Finally, the fourth stage generates fuzzy MFs from the feature coefficients calculated in stage 3, and fuses future feature coefficients generated from multiple ML models using fuzzy rules. The output of the system is the final feature importance category and degree of membership.

### Data Pre-processing

In Stage 1, it is the same as MME framework, data undergoes pre-processing to manage missing values and outliers, identify and calculate relevant features, and normalise features to reduce data bias and to ensure that the features are on similar scales to improve ML performance. Feature reduction or selection can also be performed at this stage to remove redundant, low variance, and less relevant features, thereby reducing noise in the data. The pre-processed data is then transformed to a

suitable format for training the ML models.

### Cross Validation

The pre-processed data from is randomly shuffled and partitioned into $k$ equally sized subsets for training (k-fold cross validation) [143]. For each training step, 1 subset is left out and the remaining *k-1* subsets are used to train the model. The training process is repeated $k$ times, each time leaving out a different subset and using the remaining subsets as the training data. This produces $k$ trained models. As mentioned previously, each subset of train/test data split produces different FI results and thus, provides different insights. Therefore, the cross-validation stage is crucial for capturing the variance in feature gradients and feature contributions across every combination of instances in the data used for training and testing.

### Ensemble of Machine Learning and Feature Importance Methods

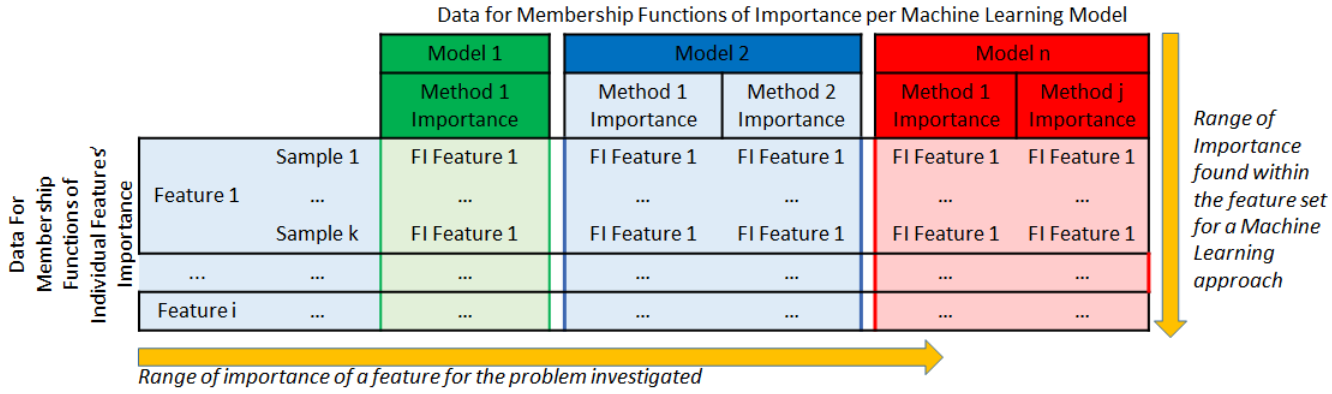



Figure 4.4: FEFI stage 3 process.

An ensemble of ML models (n models) is generated, where each model is trained

k times on the partitioned subsets of data. This produces $k * n$ trained models. Each trained model employs $j$ number of FI techniques to obtain the coefficients of importance for all the features. If each model is trained on $i$ features, the combined ML and FI methods generate a $i*j*k*n$ vector of importance coefficients. This vector encapsulates the variance in the learning characteristics and performance of the ML models and the variance in calculating importance by the FI techniques. This vector is reshaped to a suitable format for the FIS i.e., $n$ vectors of length $i*j*k$ to present the $n$ MFs for the ML models. Figure 4.4 illustrates the combined process of stage 3 and 3.

**Generating Membership Functions**

To model any disagreements or uncertainties in the importance coefficients produced by the different data subsets, ML models, and FI approaches, MFs are established from the coefficients that represent the context-dependent importance of features, and generate soft boundaries between the MFs to capture the intermediate possibilities in feature importance classification. To generate the input and output MFs, boxplot distributions of the importance coefficients generated by each ML model and the actual coefficients respectively [138] are employed. Each MF is represented by the fuzzy sets (linguistic terms); 'low', 'moderate' or 'high' importance to facilitate understanding and interpretation of feature importance for domain experts and decision makers. The five-number summary (i.e. minimum (min), first quartile (Q1), median (Q2), third quartile (Q3) and maximum (max)) of the boxplot distribution are utilised to construct the MFs. Z and S-MFs are used for 'low' and 'high' fuzzy

sets as they are most suitable for representing the far-end boundaries with maximum degrees of membership. For instance, feature coefficient of 0 is expected to be in the 'low' MF with a degree of 1 and a coefficient of 1 to be in the 'high' MF with a degree of 1. The Z-MF is represented using the minimum and median values of the distribution and the S-MF with the median and maximum values. The 'moderate' MF is represented with a triangular-MF using first quartile, median and third quartile values. Figure 4.5 shows an example of MFs generated from a boxplot distribution.

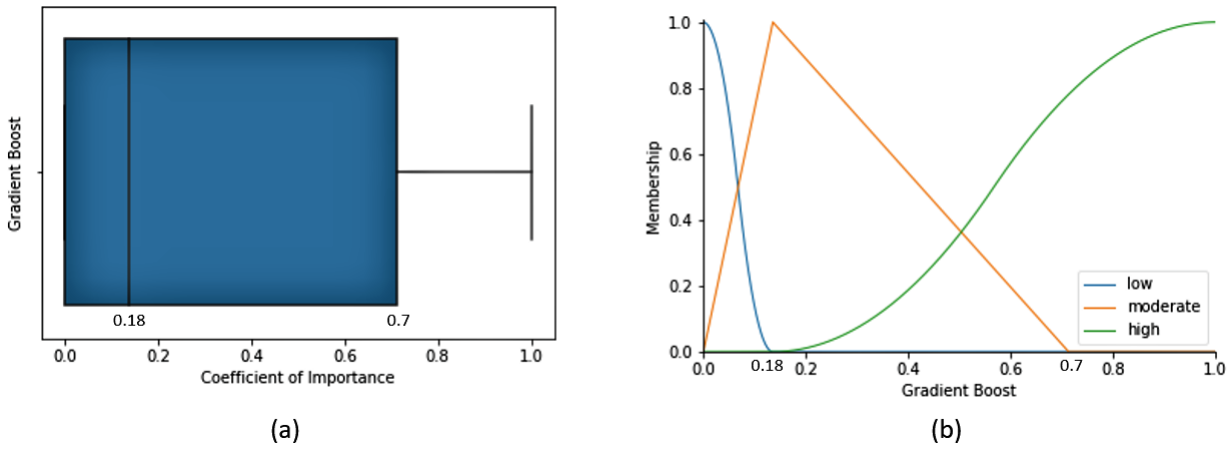



Figure 4.5: Membership functions generated from boxplot distribution: a) boxplot distribution of important values, and b) extracted MFs using our boxplot data-driven approach.

The boxplot in Figure 4.5a represents the distribution of importance coefficients for all features when GBT is trained on all data subsets and multiple FI techniques employed on the trained GBT models. By using the five-summary values of the boxplot (i.e. min=0, Q1=0, Q2=0.18, Q3=0.7 and max=1), MFs such as the ones in Figure 4.5b is generated as follows, 'low' MF = Z(0, 0.18), 'moderate' MF = triangular (0, 0.18, 0.7) and 'high' MF = S(0.18, 1). It can be observed that 50%

of the coefficients are lower than 0.18 as shown by the narrow 'low' importance MF, implying that the model has a smaller uncertainty in identifying 'low' importance features and greater uncertainty in identifying 'moderate' and 'high' importance features.

**Fuzzy rule generation and inference**

Wang-Mendel method is employed to generate the rules to combine the MFs generated by the importance coefficients. The Wang-Mendel method is one of the most commonly used rule generation technique and is most applicable for problems with ground truth data such as, the synthetic datasets used in this study. This technique uses a supervised learning approach with input-output pairs, as follows:

$$\begin{aligned}
&(FI^{1,1}_{ML_1}, FI^{1,1}_{ML_2} ... FI^{1,1}_{ML_n}; y^1),\\
&(FI^{2,1}_{ML_1}, FI^{2,1}_{ML_2} ... FI^{2,1}_{ML_n}; y^2),\\
&\quad\vdots \qquad\qquad\qquad\qquad \vdots\\
&(FI^{i,j}_{ML_1}, FI^{i,j}_{ML_2} ... FI^{i,j}_{ML_n}; y^i),
\end{aligned} \tag{4.4}$$

where $i$ is the number of features, $j$ is the number of FI techniques and $n$ is the number of ML models. Each abbreviation ($FI^{i,j}_{ML_n}$) denotes the corresponding importance coefficient of feature $i$ obtained from the $n^{th}$ ML and $j^{th}$ FI methods and $y$ denotes the actual coefficient of importance (ground truth). Table 4.1 shows an example of the IF/THEN rules generated from sample dataset called X where it has 10 features and low interactivity between features trained with GBT, RF and Extra Trees (ET),

and coefficients calculated using PI, SHAP and Gini importance techniques.

Table 4.1: Rules generated using the Wang-Mendel method on dataset X (10feat-low-inter)

| Rule No | Rule Description |
|---|---|
| 1 | **IF** GBT[‘low’] **and** RF[‘low’] **and** SVR[‘low’] **and** ET[‘low’] **THEN** output[‘low’] |
| 2 | **IF** GBT[‘high’] **and** RF[‘high’] **and** SVR[‘high’] **and** ET[‘high’] **THEN** output[‘high’] |
| 3 | **IF** GBT[‘moderate’] **and** RF[‘moderate’] **and**and SVR[‘high’] **and** ET[‘moderate’] **THEN** output[‘high’] |
| 4 | **IF** GBT[‘moderate’] **and** RF[‘moderate’] **and** SVR[‘moderate’] **and** ET[‘moderate’] **THEN** output[‘high’] |
| 5 | **IF** GBT[‘high’] **and** RF[‘high’] **and** SVR[‘moderate’] **and** ET[‘high’] **THEN** output[‘high’] |
| 6 | **IF** GBT[‘low’] **and** RF[‘low’] **and** SVR[‘moderate’] **and** ET[‘low’] **THEN** output[‘moderate’] |
| 7 | **IF** GBT[‘moderate’] **and** RF[‘low’] **and** SVR[‘low’] **and** ET[‘low’] **THEN** output[‘moderate’] |
| 8 | **IF** GBT[‘moderate’] **and** RF[‘moderate’] **and** SVR[‘low’] **and** ET[‘low’] **THEN** output[‘moderate’] |

Next, a rule-based inference method, Mamdani inference technique [141], is used to compute the rule strength and the output MFs. The rule strength is computed using the fuzzy operators AND (minimum of membership degrees), OR (maximum of membership degrees) and NOT (1-membership degree). The rule strengths determine the degree of membership for the output MFs. The output MFs describe the final FI in terms of understandable linguistic categories (i.e. ‘low’, ‘moderate’ or ‘high’ importance) and the degree of membership in each category. By using an example to demonstrate the two stages of the inference process, consider Rule 1 in Table 4.1: ***Rule 1: IF*** *GBT is ‘low’* ***AND*** *RF is ‘low’* ***AND*** *SVR is ‘low’* ***AND*** *ET is ‘low’* ***THEN*** *final FI is ‘low’*

Assuming we have the following degrees of membership in the MFs:

$$FIofGBT_{low}(A_x) = 0.15$$
$$FIofRF_{low}(A_y) = 0.2$$
$$FIofSVR_{low}(A_w) = 0.5$$
$$FIofET_{low}(A_z) = 0.3$$

Where $A_x$, $A_y$, $A_w$, and $A_z$ represent the importance coefficients of feature $A$ produced by GBT, RF, SVR, and ET, respectively. Rule 1 uses the AND operator to produce the rule strength:

$$min[GB(low)(0.15), RF(low)(0.2),$$
$$SVR(low)(0.5), ET(low)(0.3)]$$
$$min[0.15, 0.2, 0.5, 0.3] = 0.15$$

The rule strength represents the degree of membership for the 'low' output MF (the 'THEN' part of Rule 1). That is, Rule 1 produces an output importance of 'low' with a degree of 15% for feature $A$. The next step is to test the efficacy of FEFI against MME on both synthetic dataset and real world.

## 4.3 Experimental Design

To test the performance of the FEFI framework, the same case studies as the one in Chapter 3 are conducted, namely, (1) synthetic data; and (2) a real-world dataset on creep rates in LPBF. In addition to noise levels, number of features, and number of informative features, the different complexity levels of the data is added as an additional factor for the synthetic data. The complexity levels is added to further test the MME and FEFI framework under high complexity scenarios. The hypothesis of the case studies is that the FEFI framework is able to produce more accurate FI quantification compared to MME framework while accounting for the uncertainties in the underlying process and data. For the real-world dataset, the main objective is to compare FEFI to MME framework in the real world. The synthetic data case case study starts with how the training and testing data for different complexity level is generated, preprocessed, and prepared. Subsequently, the appropriate ML methods and evaluation metrics are selected for the regression tasks. After the models are trained FEFI framework is employed to generate FI quantification. Finally, the results are presented and the findings are discussed. Furthermore, as the data generation, preprocessing, ML models selection, and evaluation metric is explained in Section 3.4 for the LPBF dataset, only the results and discussions of FEFI and MME framework comparison is presented.

### 4.3.1 Summary of Data Preprocessing

In Table 4.2 a summary of the data preprocessing employed is listed.

| Task | Description |
|---|---|
| Data split ratio | 80/10/10 (train/validation/test) |
| Data split method | Randomly split between train/validation/test or Leave-one-out |
| Scaling method | All features scaled between 0-1 |
| Feature Selection | None |
| Feature Extraction (Image) | Scikit-Image library |

Table 4.2: A table of data preprocessing methods for synthetic and creep rate prediction data.

## 4.4 Case Study: Synthetic Data

### 4.4.1 Data Generation

Similar to Chapter 3, FEFI is tested on synthetic data, as it allows us to verify its efficacy against the ground truth. Three additional datasets for regression with 2000 instances and varying complexity levels. Table 4.3 shows all the datasets in this case study. The complexity of the data is determined by the strength of interaction between features. The interaction strength can be varied by changing the effective rank of the matrix, i.e. the maximum number of linearly independent feature matrix. The feature matrix rank are generated using the `make_regression` function in Python *scikit-learn* library [112]. Figure 4.6 shows the correlation matrix of the three datasets from the dataset with fully independent features to the dataset of highly correlated features. The correlation between features is calculated using Pearson correlation.

Table 4.3: Datasets with different feature interaction, number of features, features informative level, and noise's standard deviation (std) tested on the proposed Fuzzy Ensemble Feature Importance methods.

| Dataset Number | Number of features | Feature interaction | Feature informative level | Feature noise std |
|---|---|---|---|---|
| 1 | 10 | Low | 90% | 0.5 |
| 2 | 30 | Low | 90% | 0.5 |
| 3 | 50 | Low | 90% | 0.5 |
| 4 | 10 | Moderate | 90% | 0.5 |
| 5 | 10 | High | 90% | 0.5 |
| 6 | 10 | Low | 20% | 0.5 |
| 7 | 10 | Low | 50% | 0.5 |
| 8 | 10 | Low | 90% | 2.0 |
| 9 | 10 | Low | 90% | 5.0 |

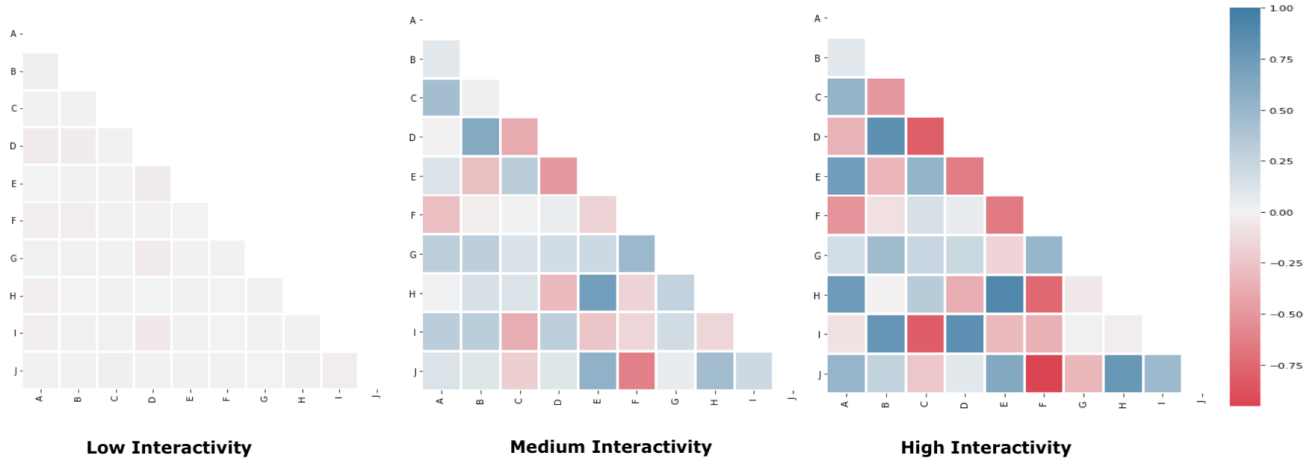



Figure 4.6: Pearson correlation matrix for datasets with different interaction strength between ten synthetic features investigated, labelled from A to J.

Additionally, different feature interaction levels are added to each dataset, as

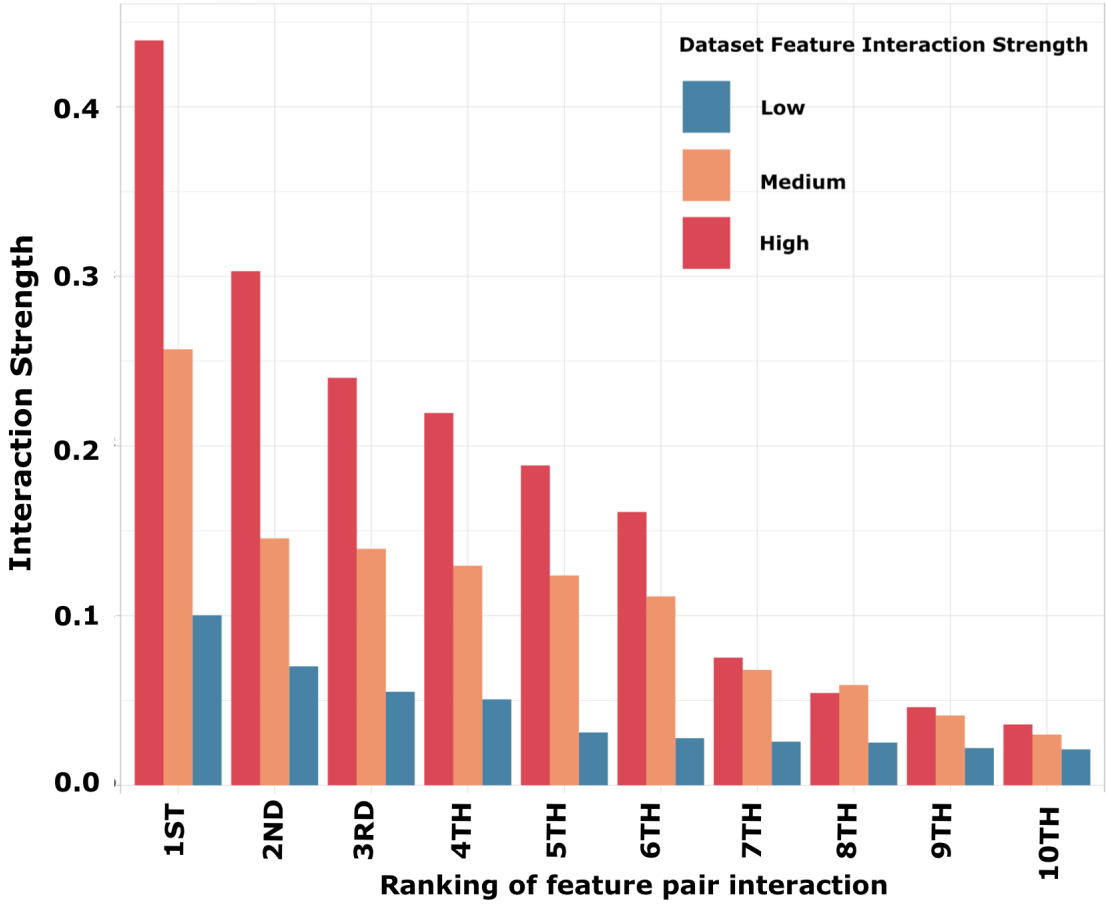



Figure 4.7: Interaction strength between pairs of features for each of the three datasets. The x-axis denotes the feature pairs with the top 10 highest interaction from each of the datasets.

illustrated in Figure 4.7. Feature interaction occurs when a feature affects one or more features, making it difficult to discern its actual feature importance value. The interaction effect between pairs of features is calculated using Friedman's H statistics [144]. Additional synthetic datasets of different properties investigated in the previous chapter are tested for consistency in experimental design and reporting of results. These include datasets of varied number of features, noise levels, and feature informative level. Feature informative levels controls how much each feature affect the output. Table 4.4 shows a summary of different datasets tested in this chapter.

Partial Dependence Plot (PDP) is employed to analyse and illustrate the relationship between the features and their outputs in Figure 4.8. The PDP is a method to show the marginal influence of one or two combined features on the output using the Monte Carlo method across the dataset [34]. The PDP in Figure 4.8 illustrates the non-linear relationship between two features and the output in high interactive setting of the synthetic dataset. The FI changes over the data space surface and the PDP plot shows the possible amount of uncertainty in the FI and synergistic importance between the features that have non-zero interactions.

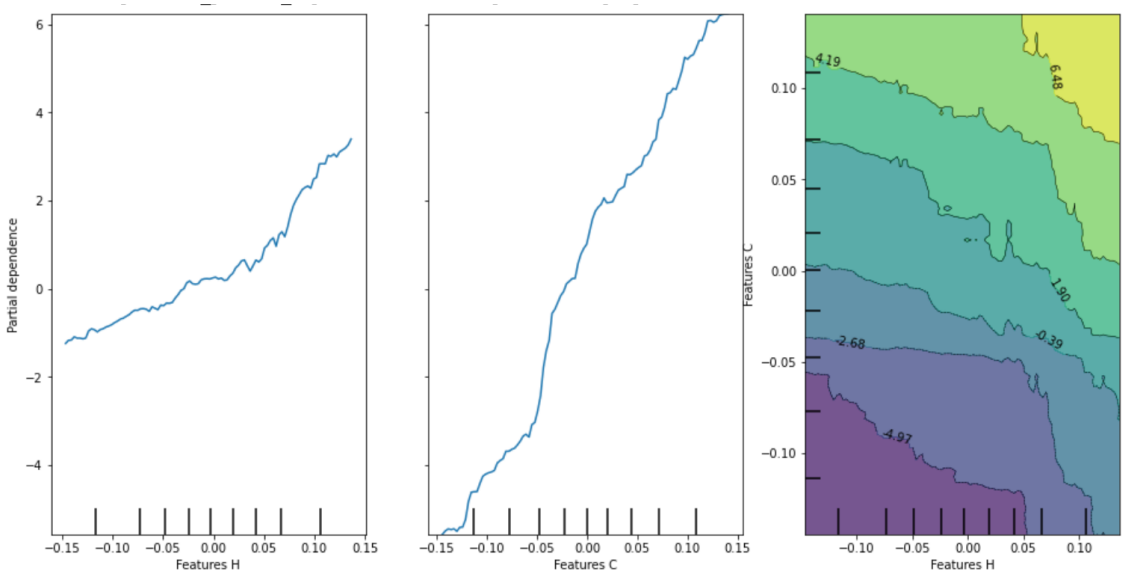



Figure 4.8: The partial dependence plot of two features on the output. The feature H (left) and the feature C (centre) plot shows the respective partial dependence of the respective feature to output. The right plot shows the combined response on the output for the feature C and H.

Table 4.4: Datasets with different feature interactions, number of features, features informative levels, and noise's standard deviation tested on the proposed Fuzzy Ensemble Feature Importance Framework.

| **Dataset properties** | **Values of properties** |
|---|---|
| Feature interaction | Low, Medium, High |
| Number of features | 10, 30, 50 |
| Features informative level | 20%, 50%, 90% |
| Noise's standard deviation | 0.5, 2.0, 5.0 |

### 4.4.2 Machine Learning Methods

Five ML models: RF, GBT, ET, SVR, MLP; and three FI methods namely, PI, SHAP and Gini Importance are implemented. We select diverse ML and FI techniques to show the benefit of FEFI in modelling the uncertainties of feature importance caused by their different learning and computation characteristics. It is important to note that our methodology can be extended to include any ML model and model-agnostic feature importance technique. The hyperparameters of each of the model are shown in Table 4.5.

Table 4.5: Hyperparameters for each of the machine learning models tested in this case study.

| Models | Hyperparameters | Values |
|---|---|---|
| Gradient Boosting Regressor | Loss | Least squares |
| | Learning rate | 0.1 |
| | Number of boosting stages | 50 |
| | Splitting criterion | Friedman MSE |
| | Minimum samples to split | 2 |
| | Maximum depth | 3 |
| Random Forest Regressor | Number of Trees | 50 |
| | Max depth | None |
| | Splitting Criterion | MSE |
| | Minimum samples to split | 2 |
| | Bootstrap | True |
| Extra Trees Regressor | Number of Trees | 50 |
| | Max depth | None |
| | Splitting Criterion | MSE |
| | Minimum samples to split | 2 |
| | Bootstrap | False |
| Support Vector Regressor | Kernel | Linear |
| | Tolerance | 0.001 |
| | Regularisation | 1.0 |
| | Epsilon | 0.1 |

| | | |
|---|---|---|
| MultiLayer Perceptron | Hidden Layer Size | 50 |
| | Activation function | ReLU |
| | Optimiser | Adam |
| | L2 Regulariser | 0.0001 |
| | Learning Rate | 0.001 |

### 4.4.3 Evaluation Metrics

The proposed FEFI is evaluated and compared to mean and majority vote MME on 4 different data properties, namely, (1) features interaction, (2) number of features, (3) features informative level, and (4) the noise level of features. Among the 4 data properties, all except feature interaction are tested in the previous chapter. Feature interaction data properties are also tested as interaction between features are common in real-world datasets. Each data property is split further into 3 different subsets i.e. whole data, train, and test sets to perform feature importance quantification. Whole dataset represents the entire data including all k subsets from cross validation. Train dataset is the k-1 fold data and test dataset is the 1 fold left out data. MAE and RMSE are chosen as the evaluation metrics to calculate the difference predicted FI and actual FI.

### 4.4.4 Results

The results of FEFI compared to the MME framework with mean and majority vote decision fusion strategy using datasets from Table 4.3.

Table 4.6 shows the errors due to calculating the final FI using the different fusion strategies as the number of features in the data increases (datasets 1, 2 and 3 in Table 4.3). Increasing the number of data features with a 90% feature informative level made it more difficult for crisp strategies (MME with mean or majority vote) to quantify feature importance. This is illustrated by the increase in FI errors from 10 to 30 features and from 10 to 50 features. This is due to the inefficiency of crisp methods in capturing the increased variation of FI coefficients produced by additional features. In contrast, FEFI exhibited minor differences in FI errors for different number of features with lower FI errors with 30 and 50 data features compared to the crisp mean and majority vote FIs. This indicates that FEFI efficiently captures the increased FI uncertainty caused by extra number of features compared to the crisp strategies.

Table 4.6: MAE and RMSE comparison between three different feature importance decision fusion approaches: (1) Mean, (2) Majority Vote, (3) Fuzzy Logic on three different number of features and three different subsets of data.

| Number of features | | Mean | | Majority Vote | | Fuzzy | |
|---|---|---|---|---|---|---|---|
| | Data subset | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| Dataset 1: 10 | Whole | 0.146 | 0.168 | **0.124** | **0.143** | 0.148 | 0.181 |
| | Train | 0.146 | 0.168 | **0.124** | **0.143** | 0.126 | 0.156 |
| | Test | 0.154 | 0.181 | **0.128** | **0.153** | 0.135 | 0.169 |
| Dataset 2: 30 | Whole | 0.379 | 0.435 | 0.397 | 0.454 | **0.107** | **0.133** |
| | Train | 0.379 | 0.434 | 0.397 | 0.454 | **0.109** | **0.135** |
| | Test | 0.377 | 0.433 | 0.398 | 0.455 | **0.117** | **0.148** |
| Dataset 3: 50 | Whole | 0.297 | 0.365 | 0.293 | 0.360 | **0.128** | **0.155** |
| | Train | 0.298 | 0.365 | 0.294 | 0.361 | **0.131** | **0.155** |
| | Test | 0.286 | 0.354 | 0.284 | 0.351 | **0.128** | **0.155** |

Table 4.7 shows errors in calculating the final FIs as the interactions between features in the data increase (datasets 1, 4 and 5 in Table 4.3). As with the number of features, there is an increase in errors for mean and majority vote crisp fusion strategies when the interactions between data features increase from low to moderate and from low to high interaction. However, there is only minor differences in errors from medium to high feature interaction for the 3 datasets. For FEFI, there is only minor differences in FI errors for different levels of feature interaction and lower FI errors for medium and high feature interaction levels compared to MME with mean and majority vote. Again, this indicates that the fuzzy approach efficiently captures the

increased variation of FI coefficients caused by higher feature interactions compared to the crisp strategies.

Table 4.7: MAE and RMSE comparison between three different feature importance decision fusion approaches: (1) Mean, (2) Majority Vote, (3) Fuzzy Logic on three different features interaction level and three different subsets of data.

| Interaction level | | Mean | | Majority Vote | | Fuzzy | |
|---|---|---|---|---|---|---|---|
| | Data subset | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| Dataset 1: Low | Whole | 0.146 | 0.168 | **0.124** | **0.143** | 0.148 | 0.181 |
| | Train | 0.146 | 0.168 | **0.124** | **0.143** | 0.126 | 0.156 |
| | Test | 0.154 | 0.181 | **0.128** | **0.153** | 0.135 | 0.169 |
| Dataset 4: Medium | Whole | 0.203 | 0.248 | 0.220 | 0.275 | **0.141** | **0.189** |
| | Train | 0.203 | 0.249 | 0.221 | 0.276 | **0.135** | **0.179** |
| | Test | 0.202 | 0.251 | 0.216 | 0.273 | **0.148** | **0.188** |
| Dataset 5: High | Whole | 0.222 | 0.288 | 0.249 | 0.314 | **0.125** | **0.164** |
| | Train | 0.222 | 0.288 | 0.250 | 0.315 | **0.139** | **0.177** |
| | Test | 0.223 | 0.291 | 0.226 | 0.288 | **0.144** | **0.180** |

Table 4.8 shows the FI errors obtained from the different fusion strategies as the informative level of features increases i.e. datasets 1, 6 and 7 in Table 4.3. It is observed that a rise in FI errors for MME mean and majority vote decision fusions as feature informative levels increase from 20% to 50%, 20% to 90% and 50% to 90%. This means the variation between the calculated crisp FI coefficients increases as the informative level of features increases. The results for FEFI show minor increase in FI RMSE while having larger increases in MAE as feature informative levels increase

from 20% to 50%, 20% to 90%, and 50% to 90%. This implies that as the number of informative features becomes available there is no outlier FI quantification by FEFI as the RMSE did not fluctuate. However, the results shows that for low number of informative features, MME with majority vote achieves lower FI quantification error compared to FEFI.

Table 4.8: MAE and RMSE comparison between three different feature importance decision fusion approach: (1) Mean, (2) Majority Vote and (3) Fuzzy Logic on three different features informative levels and three different subsets of data.

| Informative level | | Mean | | Majority Vote | | Fuzzy | |
|---|---|---|---|---|---|---|---|
| | Data subset | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| Dataset 6: 20% | Whole | 0.008 | 0.011 | **0.007** | **0.009** | 0.078 | 0.177 |
| | Train | 0.008 | 0.011 | **0.007** | **0.009** | 0.084 | 0.178 |
| | Test | 0.010 | 0.012 | **0.008** | **0.010** | 0.078 | 0.178 |
| Dataset 7: 50% | Whole | 0.078 | 0.101 | **0.061** | **0.086** | 0.111 | 0.164 |
| | Train | 0.079 | 0.102 | **0.062** | **0.088** | 0.099 | 0.144 |
| | Test | 0.093 | 0.134 | **0.078** | **0.121** | 0.111 | 0.164 |
| Dataset 1: 90% | Whole | 0.146 | 0.168 | **0.124** | **0.143** | 0.148 | 0.181 |
| | Train | 0.146 | 0.168 | **0.124** | **0.143** | 0.126 | 0.156 |
| | Test | 0.154 | 0.181 | **0.128** | **0.153** | 0.135 | 0.169 |

Table 4.9 illustrates the errors obtained from calculating the final FI as the level of noise in data increases from 0.5 noise standard deviation to 2.0 and 5.0 (datasets 1, 8 and 9 in Table 4.3). It can be observed that no significant increase in FI errors for mean, majority vote and FEFI fusion strategies as the noise standard deviation

increases from 0.5 to 2.0, 0.5 to 5.0, and 2.0 to 0.5. This means the variation or uncertainty of FI coefficients is only slightly affected by the changes of the level of noise in the data. However, FEFI produces a significantly lower FI error compared to the crisp mean and majority vote when the noise standard deviation is 5.0. This is because of its capability to effectively capture noise in data using its MFs.

Table 4.9: MAE and RMSE comparison between three different feature importance decision fusion method: (1) Mean, (2) Majority Vote, (3) Fuzzy Logic on three different features noise levels and three different subsets of data.

| Noise's Standard Deviation | | Mean | | Majority Vote | | Fuzzy | |
|---|---|---|---|---|---|---|---|
| | Data subset | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| Dataset 1: 0.5 | Whole | 0.146 | 0.168 | **0.124** | **0.143** | 0.148 | 0.181 |
| | Train | 0.146 | 0.168 | **0.124** | **0.143** | 0.126 | 0.156 |
| | Test | 0.154 | 0.181 | **0.128** | **0.153** | 0.135 | 0.169 |
| Dataset 8: 2.0 | Whole | 0.150 | 0.170 | 0.150 | 0.169 | **0.125** | **0.152** |
| | Train | 0.150 | 0.171 | 0.152 | 0.169 | **0.133** | **0.180** |
| | Test | 0.165 | 0.195 | 0.163 | 0.200 | **0.131** | **0.161** |
| Dataset 9: 5.0 | Whole | 0.151 | 0.173 | 0.155 | 0.180 | **0.117** | **0.137** |
| | Train | 0.150 | 0.172 | 0.151 | 0.171 | **0.131** | **0.168** |
| | Test | 0.164 | 0.195 | 0.165 | 0.197 | **0.117** | **0.137** |

In summary, these results show that our fuzzy FI fusion approach outperforms mean and majority vote crisp feature importance fusion methods in capturing increased variation of FI coefficients caused by increased data dimensionality, complexity, and noise . This is because FEFI explores the data space more thoroughly

as it uses multiple samples of data to make decisions about the importance of features. It also uses distributions of the data to provide better definitions of FI and soft boundaries to capture the intermediate uncertainties of FI classification.

### 4.4.5 Discussion

FEFI has the following advantages and limitations in interpreting the importance of features. It provides a clear and accurate interpretation of feature contribution to different ML model outcomes using the ML MFs. For example, Figure. 4.9 shows different interpretations for the same FI coefficient (0.2) by the MFs generated from dataset 1 (refer dataset description in Table 4.3) for GBT, RF, SVR, and MLP. Figure. 4.9 illustrates that a FI coefficient of 0.2 has a 70% likelihood of low importance for SVR and a 75% likelihood of low importance for MLP, as might be expected for a coefficient close to 0. However, for GBT and RF, a coefficient of 0.2 has a high likelihood of moderate importance and no likelihood of low importance i.e. 90% likelihood of moderate importance for GBT and 85% for RF. Therefore, without the generation of these MFs, the interpretation of the coefficients produced by the different ML approaches will be misleading. In addition, Figure 4.9 supports our use of an ensemble of diverse ML approaches to complement outputs. Different MF representations is observed for the different ML approaches but similar MFs for approaches with similar learning characteristics. For example, the low importance FS for GBT is very narrow compared SVR and MLP but very similar to that of RF due to a similar tree-based learning process.

FEFI also provides more reliable interpretations of the final FI for domain ex-

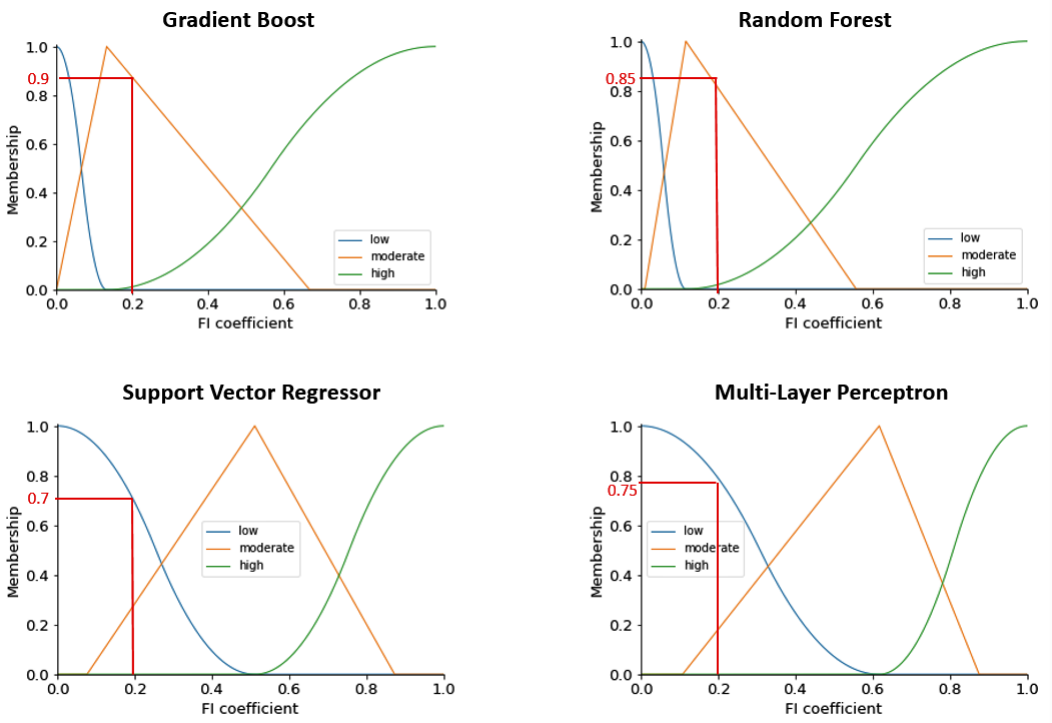



Figure 4.9: Membership functions of ML approaches generated from Dataset 1 showing different interpretations of importance for the same FI coefficient

perts. Using the output MFs as crisp FI coefficients produced by fusion strategies may be misleading without context. For example, Figure 4.10 shows the final importance coefficient (0.9) after fusing the importance obtained from the different ML approaches. Without context (the output MF), it might be assumed that 0.9 has a high likelihood of high importance as 0.9 is very close to 1 (maximum likelihood), whereas it can be observed that 0.9 has a 40% likelihood of high importance when the entire feature space is considered (indicated by the height of the shaded region in Figure 4.10). This context provided by FEFI is important in safety-critical systems such as predictive maintenance by quantifying the extent or critical level of a

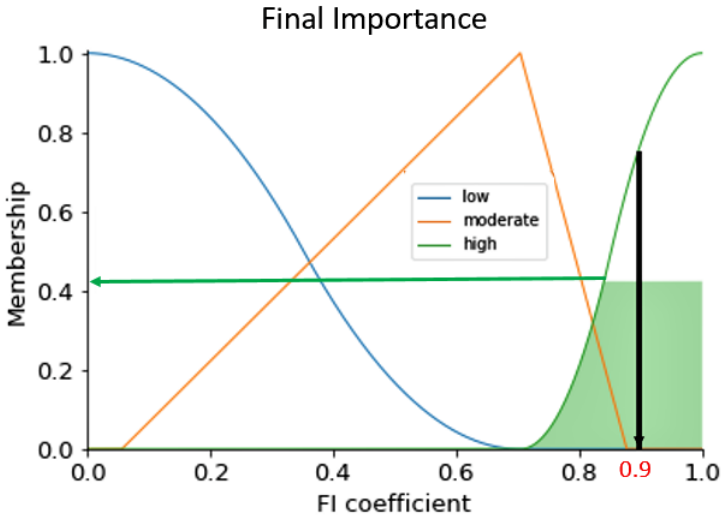



Figure 4.10: An example of the final importance of a feature after fusing the importance coefficients obtained from the different ML approaches.

feature's contribution to the decision.

Furthermore, FEFI shows great potential as a diagnostic tool in identifying data subsets with extreme cases of feature importance (this might be the case of activity cliffs in quantitative structure activity relationship modelling) or with significant variation of feature importance (e.g., for significantly nonlinear response surfaces [145]). This is achieved by producing feature MFs at different data samples or time steps to investigate the importance of a feature in the samples. For example, Figure 4.11 shows the importance of a feature generated by different data samples of dataset 1 when PI is used to calculate the feature's importance for best performing GBT model. It can be observed that the feature's importance coefficient is different in the samples; 0.8 in sample 20 with a 90% likelihood of high importance and 0.5 in sample 60 with a 30% likelihood of high importance. Real-world systems can pro-

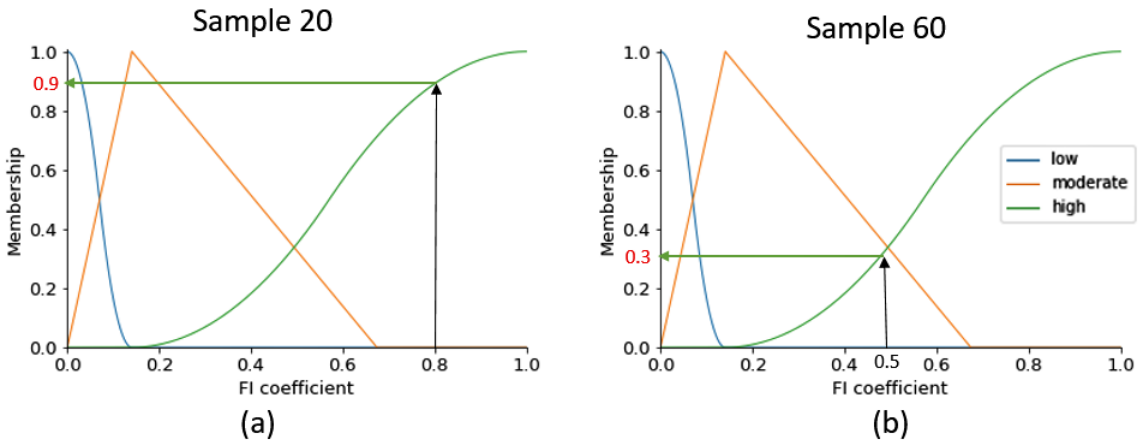



Figure 4.11: The same feature having different FI coefficients in different samples of Dataset 1. (a) Sample 20 produces a FI coefficient of 0.8, and (b) Sample 60 produces a FI coefficient of 0.5

vide more extreme cases as they produce dynamic and multifaceted data where the importance of a feature may vary in different samples due to interactions with other features or supplementary data from other data sources. For example, in predictive maintenance, when multiple sources of heterogeneous data (sensors, images, etc) are merged, features may have different importance depending on data stratification.

## 4.5 Case Study: Main Factors Affecting Creep Rates In Laser Powder Bed Fusion Using Fuzzy Ensemble Feature Importance

This case study utilises the same data and methodology as that described in Section 3.6 with the addition of employing FEFI framework to identify the importance of

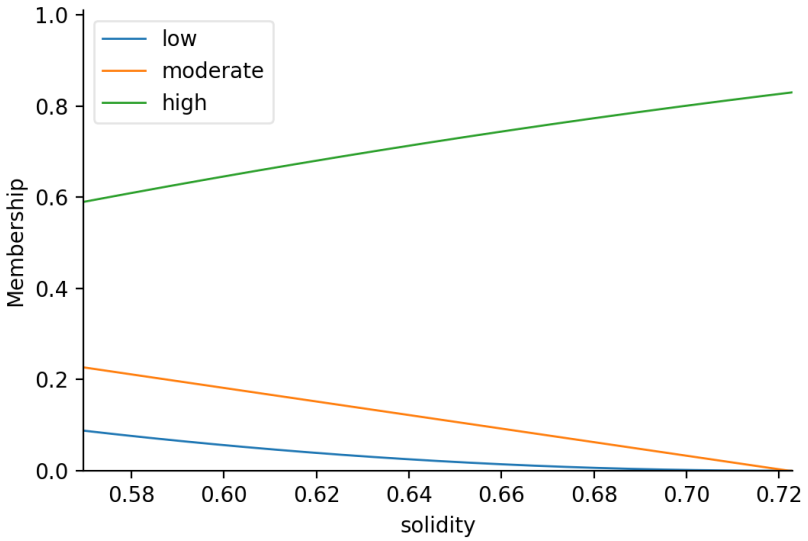
different material descriptors. The aim of this case study is to compare the difference between MME and FEFI framework on a real-world dataset.

### 4.5.1 Results

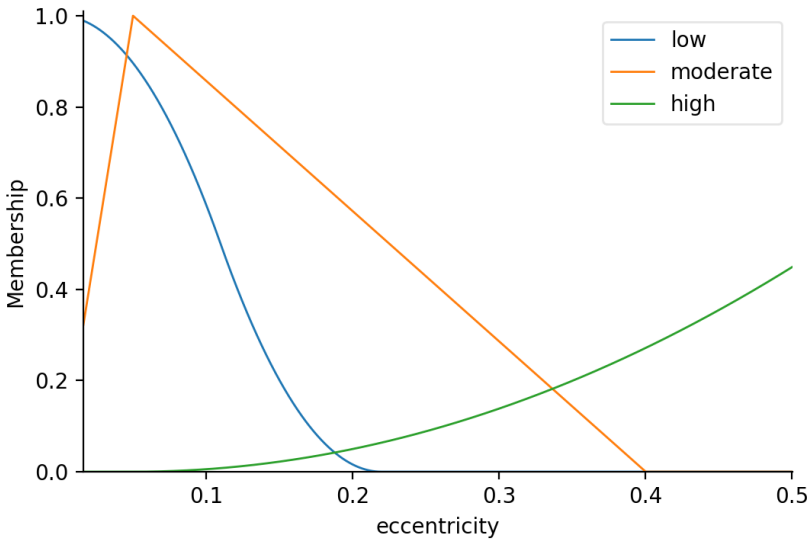
Figure 4.12 shows the ensemble material descriptor importance using MME (Non-fuzzy) and FEFI (Fuzzy) from top three most performant ML models i.e. RF, GBT, and SVR for the LOCO experiment. The four most important material descriptors identified by both MME and FEFI are the same, i.e, number of pores, solidity (density), build orientation, and scan strategy. However the rank and material descriptor importance are different between MME and FEFI. For FEFI, the top four rank from most important to least important are number of holes, density, scan strategy, and build orientation. For MME, the rank of importance is density, number of holes, build orientation, and scan strategy. Overall, both MME and FEFI's material descriptor importance ranking are similar and correlated. The importance given to each material descriptors however are different. For MME, only two material descriptors are given a importance greater than 0.8 (out of a maximum of 1.0) whereas FEFI have five material descriptors with importance greater than 0.8. Furthermore, there is a large drop in importance between the second and third most importance for MME, from 0.95 to 0.73 whereas the large drop off for FEFI happens between the fifth and the sixth most importance material descriptor, from 0.85 to 0.46. The position where the drop off happens in importance in MME compared to FEFI shows that MME is biased towards to a few material descriptors while FEFI assigned material descriptors importance more uniformly with less biases. A possible reason that FEFI

produces material descriptors importance that are more uniform is that it accounts for the overlaps in MFs (uncertainties) and therefore, importance values skew less towards the extreme value such as 0 or 1.

Additionally, one of the main features of FEFI is to produce a fuzzy prediction for each of the material descriptors importance categorised into 'Low', 'Moderate', and 'High'. In Figure 4.13a, the plot shows that the importance of density has a large membership allocated to 'High', less so for 'Moderate', and the lowest for 'Low' indicating that the material density is most likely of high importance. In Figure 4.13b, the plot shows the eccentricity having a high membership of being 'Moderate' and 'Low' importance and low membership of being 'High' importance. The membership function also shows that as the importance increases, the membership of eccentricity being an important material descriptor also increases. The membership functions for the remaining material descriptors are introduced in Appendix A.2.




(a)




(b)

Figure 4.13: Fuzzy membership of Solidity (density) and Eccentricity importance split into 'Low', 'Moderate', and 'High'.

### 4.5.2 Discussions

The results from Figure 4.12 shows that that the top factors influencing the creep rate were the density, number of pores, build orientation and scan strategy for both MME and FEFI. The effect of density and number of holes on creep rate are well documented [146, 147, 148] which shows that both MME and FEFI were correctly identifying the main causes for the creep behaviour. However, MME and FEFI disagrees on the two subsequent important material descriptors, scan strategy and build orientation whereby FEFI assigned higher importance to both factors especially scan strategy while MME gave a lower importance score. FEFI's importance assignment for scan strategy is in accordance to the findings in literature where scan strategy contributes considerably to creep behaviour. For example, a study found residual porosity to be mostly present in scan strategies such as the Stripe strategy, due to excessive energy density in laser overlap regions [149]. Similarly, build orientation that was given a higher importance by FEFI compared MME, is known to be an important factor that contributes to creep rate. Build orientation has an effect on the creep rate due to the resulting microstructure it produces [150]. FEFI's ability in correctly identifying the importance for scan strategy shows that it is more accurate and robust compared to MME in real-world dataset. The likely reason that FEFI outperforms MME in this particular case is that it is able to incorporate experts' knowledge on the material descriptors that might contribute to the creep behaviour whereas MME rely only on the information generated from data, ML models, and feature importance technique without domain knowledge. Overall, FEFI is more accurate than MME at identifying the correct factors that affects the creep rate while

providing linguistic description to aid stakeholders in understanding the importance of different material descriptors. Additionally, FEFI fundamentally incorporates uncertainty estimation in the form of membership functions that also assist in stake holder's decision making.

## 4.6 Summary

In summary, this chapter discusses the shortcomings of MME framework approach and as a result, proposed an improvement by incorporating fuzzy system to create a new FEFI framework. FEFI has the ability to capture the uncertainties that arises from the feature importance explanation while resenting the feature importance in linguistic term. Furthermore, by incorporating the uncertainties in its FI quantification FEFI is able to retain useful information that would otherwise not be accounted for during decision fusion of MME framework. To compare the performance of MME and FEFI framework, they both tested using the same case study from previous chapter. The results shows that FEFI is able to quantify the FI more accurately while providing the FI values in linguistic terms.

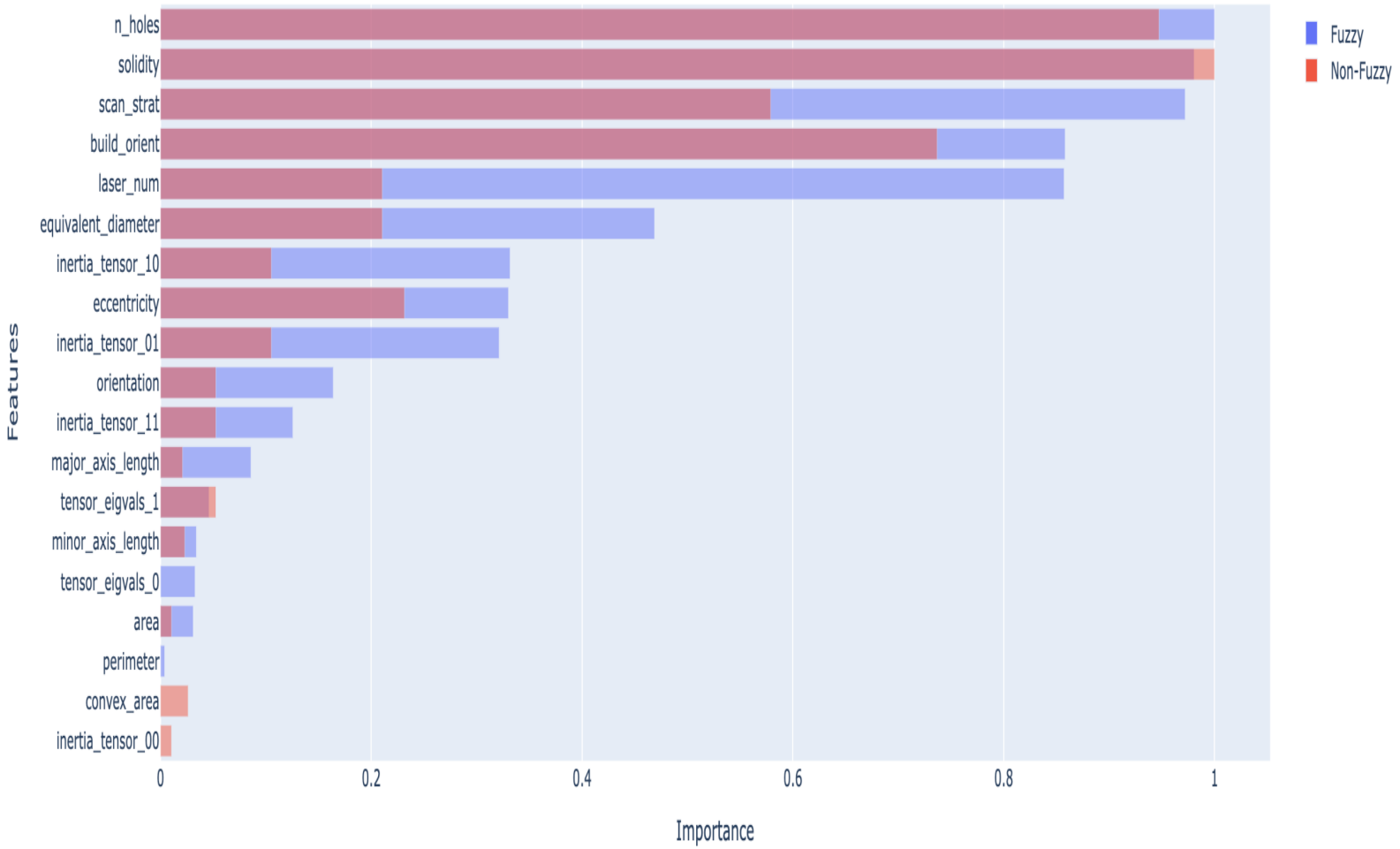



Figure 4.12: Ensemble material descriptor importance for Leave-One-Case-Out experiment.

# Chapter 5

# Conclusions

## 5.1 Conclusions

In recent years CBM technology has become increasingly popular and are rapidly adopted by various industries such as aerospace and manufacturing. The data from the utilisation of CBM has enabled expensive and essential assets to be continuously monitored. Using the collected data, one can create tools to predict and manage the availability and reliability of their assets. The prediction and reliability of these machines fall under the studies of predictive maintenance. Traditionally, predictive maintenance is accomplished using statistical techniques, such as Weibull or logistic regression to model RUL or predict fault. However, the data collected for predictive maintenance are generally large and complex due to the high dimensionality of multi-sensors, thus making ML and, more specifically, DL, an ideal candidate to handle the data. However, current research trends towards experimenting with different model

architectures to improve predictive performance without utilising domain-specific knowledge. Furthermore, as predictive maintenance is often practised in safety-critical systems, such as automotive and aerospace that emphasise the interpretation of models for regulation, ethical concerns, and general safety, more research must focus on it. This thesis presented the inclusion of domain knowledge in the ML systems through the introduction of dynamic weighted and asymmetric loss functions to improve the ML models' predictive performance and created two frameworks that accurately and safely extracts knowledge from the trained ML models to support post-training interpretability.

The main findings of dynamic weighted and asymmetric loss functions was it improves the DL models predictive performance improving both RUL and fault detection accuracy. It was shown through the CMAPSS and APS case studies that by incorporating implicit knowledge or external domain knowledge the learning models were able to be more biased towards predictions that more favourable for predictive maintenance such as early prediction for RUL or false positive prediction rather than false negative prediction in fault detection. However, there are additional work incur by using dynamically weighted and asymmetric loss functions due to extra parameters that require tuning. In the case of extracting knowledge from learned ML and DL models, the two proposed framework, namely, MME and FEFI framework were able to more accurately quantify the global FI. The MME framework was initially proposed to overcome the inadequacies of single method ensemble — whereby only a single FI method is used. To extend the framework further to include uncertainty quantification and linguistic explanation fuzzy system was used to create the FEFI

framework. By incorporating fuzzy system into the framework, it was able to quantify global FI more accurately compared to MME framework and single method ensemble on both synthetic and real-world creep rate prediction dataset. Additionally, the FEFI framework was able to take in account of the uncertainty produced at each step of the process and incorporate the information in the final decision fusion steps. Moreover, it was able to provide simple linguistic explanation to the FI quantification that could help stakeholder in making more informed decisions. Overall, by using dynamic weighted and asymmetric loss functions and also the FEFI framework to include and extract domain knowledge into learning models, it improves the overall learning along with decision making process that ultimately lead to more responsible and safer usage of Artificial Intelligence in predictive maintenance.

### 5.1.1 Summary

In Chapter 2, a dynamically weighted loss function method for regression is introduced to automatically extract the knowledge of which particular instances the model should pay more attention during its learning process. The main objectives of this chapter are to develop and test supervised DL models with a new type of knowledge embedded loss functions to improve predictive capabilities for sensor-based data by favouring early prediction in RUL and subsequently asses the advantage of knowledge embedded loss functions on RUL of gas turbine engine and fault detection in air pressure systems. The proposed weighted loss function is tested on a aerospace RUL dataset and showed improved performance in RUL prediction compared to standard loss functions. Additionally, FL, a variant of dynamically weighted loss functions for

classification task is also tested on a air pressure system fault detection problem. Similar to the proposed dynamically weighted loss function, FL showed improved fault detection accuracy. Interestingly, the standard symmetrical metrics such as MSE and RMSE generally show minor improvement for both dynamically weighted loss functions but larger improvement when calculated through predictive maintenance related metrics. Larger improvement over predictive maintenance related metrics indicates that the ML models are learning to be biased towards outcome that are often desirable in predictive maintenance such as favouring early rather than late prediction for RUL. However, one disadvantage of the proposed loss function is difficulty in tuning the the weights that is given instances that requires extra attention. Biasing the model too much or too little can lead to outcomes that under performs compared standard loss function. Furthermore, in this chapter a asymmetrical loss function for regression problem is also introduced. The asymmetrical loss function is embedded with the knowledge of favouring early rather than late predictions for RUL task. The difference between the proposed dynamically weighted loss function and the proposed asymmetrical loss function is that the asymmetrical loss function is explicitly biased through knowledge embedding in the loss functions while the dynamically weighted loss function learns the bias through data. An advantage of the asymmetrical loss function is that the biasing of model is not dependent of the data quality. Furthermore, precise and specific encoding of knowledge can be added into the loss function. The asymmetrical loss function is tested on the same aerospace RUL dataset and shows that it is able to achieve higher RUL prediction accuracy compared to dynamically weighted loss function. However, asymmetrical

loss function also suffers the problem where the tuning of parameters have to be precise to avoid under or over biasing the ML model. An additional disadvantage of the weighted and asymmetric loss functions is the lack of theoretical foundations with mathematical proofs of its behaviour. Furthermore, additional case should be conducted to validate the improvement of weighted loss functions and asymmetric loss functions.

In Chapter 3, the focus is shifted from embedding knowledge to improve the model to answering the question such as "How to safely and accurately interpret the results of ML models?". The objectives of this chapter are to develop and test an ensemble FI framework for decision fusion that is more accurate in FI estimates compared than the existing FI method and to develop an understanding of how ensemble FI behaves under varied data conditions.. Traditionally, the extraction of knowledge or interpretation of the outputs of trained models relies on a single feature importance method. Furthermore the pros and cons of different feature importance methods are not fully explored and if there is disagreements between the methods it is unclear which is more accurate. Therefore, a MME framework has been developed to ensemble the outputs of multiple feature importance methods from multiple ML models to provide a more accurate interpretation of feature importance. Using synthetic data as a case study, the results of MME were shown to be more accurate and robust at identifying feature importance compared to single feature importance method. Furthermore, the MME framework is tested on a creep rate prediction case study on AM material with the aim of identifying important material descriptors that led to different creep rates. While the MME are able to output

reasonably accurate material descriptor importance it is not obvious to the domain experts as it lacks further insights. Additionally, the output from MME consists of arbitrary numerical values of importance that does not have intrinsic meaning associated to the features. An appropriate solution is to attached meaning values and linguistic term to the feature importance for easier and intuitive understanding of the importance. Furthermore, MME is not able to incorporate prior domain knowledge that could improves the interpretation accuracy of feature importance.

In Chapter 4, MME framework is extended to include Fuzzy system (called the FEFI framework) to address the shortcomings of MME. The main objectives achieved in this chapter are developing a fuzzy-based method to capture the uncertainty within ensemble FI framework to accurately report explain ML models. Additionally, explained the FI in linguistic term for straightforward explanation. By incorporating Fuzzy system, not only can the FEFI framework take advantage of the accuracy and robustness improvement from ensembling multiple feature importance methods with multiple ML models it also showed the uncertainties of each feature importance. Incorporating the uncertainties of each feature importance makes the overall decision making process safer and more ethical as it provides a more complete view of the output for decision makers. Furthermore, the FEFI frameworks is able to incorporate external knowledge as prior information into its system resulting in more accurate feature importance accuracy. The FEFI framework is tested on synthetic data to provide quantitative evidence that it is more accurate than the MME framework under different data conditions. Furthermore, it is also tested on the same AM materials creep prediction dataset in Chapter 3 and FEFI showed to be more accurate

than MME in identifying the importance for each material descriptors while providing linguistic explanation term for decision makers. A shortcomings of this chapter is the lack of comparison of between different approach to capture uncertainties in the different ML models and FI approaches. Methods such as bayesian statistics is also a viable way to capture the uncertainties while incorporating prior knowledge. However, due to time constraint this investigation was not possible. Furthermore, a deeper theoretical analysis of the FEFI framework is required to fully understands its behaviour and the limits of its performance under different data conditions. An additional improvement to the empirical analysis of this chapter could come from a better designed synthetic data. While the synthetic data used in this chapter attempts to simulate real-world data there are some conditions such as missing or censored data that are not accounted for in the analysis of this chapter.

### 5.1.2 Future Works

Some possible future works based on the research in this thesis are listed as follows:

- A theoretical analysis on the behaviour of dynamically weighted loss functions and the asymmetrical loss functions. While this thesis has developed the newly improved loss functions and showed that it performed well compared to traditional method in different case studies there is room to further understand the behaviour of these loss functions from a more foundational and theoretical point of view. A theoretical understanding of the dynamically weighted loss functions and asymmetrical loss functions can provide a complete understanding towards the different behaviour of the loss functions, which is important

when employed on safety-critical predictive maintenance systems.

- A theoretical analysis of the MME and FEFI framework could bring further understanding as to how the framework fundamentally behaves under varied conditions. Similar to the theoretical analysis of the proposed loss functions, a theoretical understanding of the MME and FEFI framework is important for employment in safety-critical predictive maintenance systems.

- A comparison of FEFI framework to other uncertainty quantification approach such as bayesian statistics. Currently, it is not obvious if FEFI is the best approach for a bayesian-based ensemble feature feature importance. On a high level, similar to Fuzzy Logic, bayesian statistics can incorporate domain knowledge through statistical priors and quantify uncertainties. Further research is required to understand which approach is more suitable for a safer way to interpret the output of ML models.

- FEFI framework provides information as to which features are important. The importance of features can serve as an indicator for the model to focus more on features that are important through the loss functions. The connection between the FEFI framework and loss functions provides and end-to-end learning paradigm that can potentially lead to more accurate predictions from model.

### 5.1.3 Publications

1. Chapter 1.

- Rengasamy, Divish, Hervé P. Morvan, and Grazziela P. Figueredo. "Deep learning approaches to aircraft maintenance, repair and overhaul: a review." 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018. — This paper provides an literature review of the approaches of DL employed in the aerospace predictive maintenance space, where gaps and opportunities for our research have been identified.

2. Chapter 2.

    - Rengasamy, D., Jafari, M., Rothwell, B., Chen, X., & Figueredo, G. P. (2020). "Deep learning with dynamically weighted loss function for sensor-based prognostics and health management". Sensors, 20(3), 723. — This paper proposes the regression-based implicit knowledge embedded loss function and introduces FL function for classification-based predictive maintenance application. Both implicit knowledge embedded loss functions are tested on RUL of gas turbine engine and fault detection in air pressure system respectively.
    - Rengasamy, Divish, Benjamin Rothwell, and Grazziela P. Figueredo. "Asymmetric loss functions for deep learning early predictions of remaining useful life in aerospace gas turbine engines." 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020. — This paper proposes the asymmetric loss functions where the loss function is embedded with knowledge of early prediction is more favourable compared to late prediction in RUL. The proposed asymmetric loss function is tested against traditional

loss functions for RUL in gas turbine engine.

- Victoria Bell, Grazziela Figueredo, Divish Rengasamy, and Benjamin Rothwell. "Anomaly Detection for Unmanned Aerial Vehicle Data Using a Stacked Recurrent Autoencoder Method with Dynamic Thresholding" 2022 *Preprint.* — This papers extends the regression-based dynamically loss functions to anomaly detection problem in predictive maintenacne with the integration of dynamic thresholding.

3. Chapter 3.

   - Rengasamy, Divish, Benjamin C. Rothwell, and Grazziela P. Figueredo. 2021. "Towards a More Reliable Interpretation of Machine Learning Outputs for Safety-Critical Systems Using Feature Importance Fusion" Applied Sciences 11, no. 24: 11854. - This paper introduces the crisp-based ensemble FI framework. The proposed framework is tested on multiple decision fusion approach and finally compared against traditional FI approaches on a synthetic dataset.
   - Sanchez, S., Rengasamy, D., Hyde, C. J., Figueredo, G. P.,& Rothwell, B. (2021). "Machine learning to determine the main factors affecting creep rates in laser powder bed fusion". Journal of Intelligent Manufacturing, 1-21. — This paper utilises the crisp-based ensemble FI framework to determine the important features contributing to the creep rate in additively manufactured alloy.

4. Chapter 4.

- Rengasamy, D., Mase, J. M., Torres, M. T., Rothwell, B., Winkler, D. A., & Figueredo, G. P. (2021). "Mechanistic Interpretation of Machine Learning Inference: A Fuzzy Feature Importance Fusion Approach". arXiv preprint arXiv:2110.11713. — This paper addresses the limitations of crisp-based ensemble FI by using fuzzy systems to capture the uncertainties of FI while producing FI explantion in linguistic terms. Furthermore, the fuzzy-based ensemble FI framework is compared against crisp-based ensemble FI framework using synthetic data.

# Bibliography


[1] Department of Business Innovation and Skills. *UK Aerospace Maintenance, Repair, Overhaul; Logistics Industry Analysis*. Feb. 2016. URL: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/502588/bis-16-132-uk-mrol-analysis.pdf.

[2] Christian Steensland Børresen. "A framework for cost-benefit analysis on use of condition based maintenance in an IO perspective". MA thesis. Norges teknisk-naturvitenskapelige universitet, Fakultet for ..., 2011.

[3] Giorgio Barone and Dan M Frangopol. "Life-cycle maintenance of deteriorating structures by multi-objective optimization involving reliability, risk, availability, hazard and cost". In: *Structural Safety* 48 (2014), pp. 40–50.

[4] Bill Read. *Digital takeover*. https://www.aerosociety.com/news/digital-takeover. Accessed: 2020-11-03.

[5] Deepam Goyal and BS Pabla. "Condition based maintenance of machine tools—A review". In: *CIRP Journal of Manufacturing Science and Technology* 10 (2015), pp. 24–35.

[6] Thyago P Carvalho et al. “A systematic literature review of machine learning methods applied to predictive maintenance”. In: *Computers & Industrial Engineering* 137 (2019), p. 106024.

[7] Divish Rengasamy, Hervé P Morvan, and Grazziela P Figueredo. “Deep learning approaches to aircraft maintenance, repair and overhaul: A review”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 150–156.

[8] The University of Texas System Administration. *Special Review of Procurement Procedures Related to the M.D. Anderson Cancer Center Oncology Expert Advisor Project*. 2017. URL: https://www.utsystem.edu/sites/default/files/documents/UT%20System%20Administration%20Special%20Review%20of%20Procurement%20Procedures%20Related%20to%20UTMDACC%20Oncology%20Expert%20Advisor%20Project/ut-system-administration-special-review-procurement-procedures-related-utmdacc-oncology-expert-advis.pdf.

[9] Lee Timothy. *Autopilot was active when a Tesla crashed into a truck, killing driver*. https://arstechnica.com/cars/2019/05/feds-autopilot-was-active-during-deadly-march-tesla-crash/. Accessed: 2020-11-03.

[10] Conrad Hal Waddington. *OR in World War 2: Operational Research against the U-boat*. Elek, 1973.

[11] Jianhui Luo et al. “Model-based prognostic techniques [maintenance applications]”. In: *Proceedings AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference.* Ieee. 2003, pp. 330–340.

[12] Tiedo Tinga and Richard Loendersloot. “Physical model-based prognostics and health monitoring to enable predictive maintenance”. In: *Predictive Maintenance in Dynamic Systems*. Springer, 2019, pp. 313–353.

[13] Marcia Baptista et al. “Forecasting fault events for predictive maintenance using data-driven techniques and ARMA modeling”. In: *Computers & Industrial Engineering* 115 (2018), pp. 41–53.

[14] Yu Hui Lui et al. “Physics-Based State of Health Estimation of Lithium-Ion Battery Using Sequential Experimental Design”. In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Vol. 51760. American Society of Mechanical Engineers. 2018, V02BT03A061.

[15] Peter S Maybeck. “Multiple model adaptive algorithms for detecting and compensating sensor and actuator/surface failures in aircraft flight control systems”. In: *International Journal of Robust and nonlinear control* 9.14 (1999), pp. 1051–1070.

[16] L Van Eykeren and QP Chu. “Sensor fault detection and isolation for aircraft control systems by kinematic relations”. In: *Control Engineering Practice* 31 (2014), pp. 200–210.

[17] ID Serna-Suárez et al. “Impact of static load models on the power distribution fault location problem”. In: *2010 IEEE/PES Transmission and Distribution Conference and Exposition: Latin America (T&D-LA)*. IEEE. 2010, pp. 706–711.

[18] Matthew Daigle, Michael Foygel, and Vadim Smelyanskiy. “Model-based diagnostics for propellant loading systems”. In: *2011 Aerospace Conference*. IEEE. 2011, pp. 1–11.

[19] Andrew KS Jardine, Daming Lin, and Dragan Banjevic. “A review on machinery diagnostics and prognostics implementing condition-based maintenance”. In: *Mechanical systems and signal processing* 20.7 (2006), pp. 1483–1510.

[20] Ioanna Aslanidou et al. “Micro Gas Turbines in the Future Smart Energy System: Fleet Monitoring, Diagnostics, and System Level Requirements”. In: *Frontiers in Mechanical Engineering* 7 (2021).

[21] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[22] Wenbin Wang. “A two-stage prognosis model in condition based maintenance”. In: *European journal of operational research* 182.3 (2007), pp. 1177–1187.

[23] Alireza Ghasemi, S Yacout, and MS Ouali. “Optimal condition based maintenance with imperfect information and the proportional hazards model”. In: *International journal of production research* 45.4 (2007), pp. 989–1012.

[24] Eric Allen Strong. *Methods and systems for predicting a remaining useful life of a component using an accelerated failure time model*. US Patent App. 16/149,668. Feb. 2020.

[25] Michael Schemper. "Cox analysis of survival data with non-proportional hazard functions". In: *Journal of the Royal Statistical Society: Series D (The Statistician)* 41.4 (1992), pp. 455–465.

[26] Woo Jung Kim et al. "Cox Proportional Hazard Regression Versus a Deep Learning Algorithm in the Prediction of Dementia: An Analysis Based on Periodic Health Examination". In: *JMIR medical informatics* 7.3 (Aug. 2019).

[27] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[28] Harris Drucker et al. "Support vector regression machines". In: *Advances in neural information processing systems*. 1997, pp. 155–161.

[29] Zhi Lv et al. "Prognostics health management of condition-based maintenance for aircraft engine systems". In: *2015 IEEE Conference on Prognostics and Health Management (PHM)*. IEEE. 2015, pp. 1–6.

[30] Yanghui Tan et al. "A one-class SVM based approach for condition-based maintenance of a naval propulsion plant with limited labeled data". In: *Ocean Engineering* 193 (2019), p. 106592.

[31] GG Acosta, CJ Verucchi, and ER Gelso. "A current monitoring system for diagnosing electrical failures in induction motors". In: *Mechanical Systems and Signal Processing* 20.4 (2006), pp. 953–965.

[32] Hack-Eun Kim et al. “Machine prognostics based on health state estimation using SVM”. In: *Asset Condition, Information Systems and Decision Models*. Springer, 2012, pp. 169–186.

[33] Tin Kam Ho. “Random decision forests”. In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282.

[34] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[35] Weizhong Yan. “Application of random forest to aircraft engine fault diagnosis”. In: *The Proceedings of the Multiconference on” Computational Engineering in Systems Applications”*. Vol. 1. IEEE. 2006, pp. 468–475.

[36] Fei Li et al. “A light gradient boosting machine for remaining useful life estimation of aircraft engines”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3562–3567.

[37] Chuan Li et al. “Gearbox fault diagnosis based on deep random forest fusion of acoustic and vibratory signals”. In: *Mechanical Systems and Signal Processing* 76 (2016), pp. 283–293.

[38] Zaharah Allah Bukhsh et al. “Predictive maintenance using tree-based classification techniques: A case of railway switches”. In: *Transportation Research Part C: Emerging Technologies* 101 (2019), pp. 35–54.

[39] Alexandra Khalyasmaa, Mihail Senyuk, and Stanislav Eroshenko. "Analysis of the state of high-voltage current transformers based on gradient boosting on decision trees". In: *IEEE Transactions on Power Delivery* (2020).

[40] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[41] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology press, 1995.

[42] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[43] Lucas Beyer et al. "Are we done with imagenet?" In: *arXiv preprint arXiv:2006.07159* (2020).

[44] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html`.

[45] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[46] Arka Daw et al. "Physics-guided neural networks (pgnn): An application in lake temperature modeling". In: *arXiv preprint arXiv:1710.11431* (2017).

[47] Prashant Kumar and Ananda Shankar Hati. "Deep convolutional neural network based on adaptive gradient optimizer for fault detection in SCIM". In: *ISA transactions* (2020).

[48] Marcia Baptista et al. "Using Explainable Artificial Intelligence to Interpret Remaining Useful Life Estimation with Gated Recurrent Unit". In: (Apr. 2020). DOI: 10.13140/RG.2.2.27721.36963.

[49] Divish Rengasamy, Benjamin Rothwell, and Grazziela Figueredo. "Towards a More Reliable Interpretation of Machine Learning Outputs for Safety-Critical Systems using Feature Importance Fusion". In: *arXiv preprint arXiv:2009.05501* (2020).

[50] Kwok L. Tsui et al. "Prognostics and Health Management: A Review on Data Driven Approaches". In: *Mathematical Problems in Engineering* 2015 (2015).

[51] Divish Rengasamy, Hervé P Morvan, and Grazziela P Figueredo. "Deep learning approaches to aircraft maintenance, repair and overhaul: a review". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 150–156.

[52] Grazziela P Figueredo, Kayode Owa, and Robert I John. *Multi-Objective Optimization for Preventive Maintenance in Transportation: A Review*. Tech. rep. University of Nottingham, 2018.

[53] Samir Khan and Takehisa Yairi. "A review on the application of deep learning in system health management". In: *Mec Sys and Sig Proc* 107 (2018). ISSN: 0888-3270.

[54] A. L. Ellefsen et al. "A Comprehensive Survey of Prognostics and Health Management Based on Deep Learning for Autonomous Ships". In: *IEEE Transactions on Reliability* 68.2 (June 2019), pp. 720–740. ISSN: 0018-9529.

[55] Peter J Huber. "Robust estimation of a location parameter". In: *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[56] Solomon Kullback and Richard A Leibler. "On information and sufficiency". In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[57] T. Lin et al. "Focal Loss for Dense Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324.

[58] Kapil Katyal et al. "Uncertainty-aware occupancy map prediction using generative networks for robot navigation". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 5453–5459.

[59] Yating Wang and Guang Lin. "Efficient deep learning techniques for multiphase flow simulation in heterogeneous porousc media". In: *Journal of Computational Physics* 401 (2020), p. 108968.

[60] Erin McGowan, Vidita Gawade, and Weihong Guo. "A Physics-Informed Convolutional Neural Network with Custom Loss Functions for Porosity Prediction in Laser Metal Deposition". In: *Sensors* 22.2 (2022), p. 494.

[61] Thomas Lofthouse. "The Taguchi loss function". In: *Work Study* (1999).

[62] Divya Pandey, Makarand S Kulkarni, and Prem Vrat. "A methodology for simultaneous optimisation of design parameters for the preventive maintenance and quality policy incorporating Taguchi loss function". In: *International Journal of Production Research* 50.7 (2012), pp. 2030–2045.

[63] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[64] Zehai Gao et al. "Deep quantum inspired neural network with application to aircraft fuel system fault diagnosis". In: *Neurocomputing* 238 (2017), pp. 13–23.

[65] Moritz Hardt and Benjamin Recht. "Patterns, predictions, and actions: A story about machine learning". In: *arXiv preprint arXiv:2102.05242* (2021).

[66] Jason D Lee et al. "Gradient descent only converges to minimizers". In: *Conference on learning theory*. PMLR. 2016, pp. 1246–1257.

[67] Sashank Reddi et al. "A generic approach for escaping saddle points". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 1233–1242.

[68] Y LeCun *et al.* "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Comp* 1.4 (1989), pp. 541–551.

[69] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667.

[70] L. C. Jain and L. R. Medsker. *Recurrent Neural Networks: Design and Applications*. 1st. USA: CRC Press, Inc., 1999. ISBN: 0849371813.

[71] Kyunghyun Cho *et al.* "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014).

[72] A Saxena *et al.* "Damage propagation modeling for aircraft engine run-to-failure simulation". In: *I Conf on Prog and Health Man*. Oct. 2008.

[73] K. Goebel et al. “Modeling Propagation of Gas Path Damage”. In: *2007 IEEE Aerospace Conference*. Mar. 2007, pp. 1–8.

[74] Lorenzo Beretta and Alessandro Santaniello. “Nearest neighbor imputation algorithms: A critical evaluation”. In: *BMC Medical Informatics and Decision Making* 16 (July 2016). DOI: 10.1186/s12911-016-0318-z.

[75] Takaya Saito and Marc Rehmsmeier. “The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets”. In: *PLOS ONE* 10.3 (Mar. 2015), pp. 1–21. DOI: 10.1371/journal.pone.0118432.

[76] Yarin Gal and Zoubin Ghahramani. “A Theoretically Grounded Application of Dropout in Recurrent Neural Networks”. In: (2016). Ed. by D. D. Lee et al., pp. 1019–1027.

[77] Sanjit A Seshia, Dorsa Sadigh, and S Shankar Sastry. “Towards verified artificial intelligence”. In: *arXiv preprint arXiv:1606.08514* (2016).

[78] Miles Brundage et al. “Toward trustworthy AI development: mechanisms for supporting verifiable claims”. In: *arXiv preprint arXiv:2004.07213* (2020).

[79] Michelle Pham et al. “Asilomar survey: researcher perspectives on ethical principles and guidelines for BCI research”. In: *Brain-Computer Interfaces* 5.4 (2018), pp. 97–111.

[80] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[81] Amina Adadi and Mohammed Berrada. "Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI)". In: *IEEE Access* 6 (2018), pp. 52138–52160.

[82] David Gunning. "Explainable artificial intelligence (xai)". In: *Defense Advanced Research Projects Agency (DARPA), nd Web* 2 (2017).

[83] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115.

[84] Mattia Carletti et al. "Explainable machine learning in industry 4.0: Evaluating feature importance in anomaly detection to enable root cause analysis". In: *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE. 2019, pp. 21–26.

[85] Prashant Kumar and Ananda Shankar Hati. "Deep convolutional neural network based on adaptive gradient optimizer for fault detection in SCIM". In: *ISA transactions* 111 (2021), pp. 350–359.

[86] Chang Woo Hong et al. "Explainable Artificial Intelligence for the Remaining Useful Life Prognosis of the Turbofan Engines". In: *202020 3rd IEEE International Conference on Knowledge Innovation and Invention (ICKII)*. IEEE. 2020, pp. 144–147.

[87] Maciej Szelażek et al. "Towards the Modeling of the Hot Rolling Industrial Process. Preliminary Results". In: *International Conference on Intelligent Data Engineering and Automated Learning*. Springer. 2020, pp. 385–396.

[88] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "" Why should I trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[89] Oscar Serradilla et al. "Interpreting Remaining Useful Life estimations combining Explainable Artificial Intelligence and domain knowledge in industrial machinery". In: *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2020, pp. 1–8.

[90] Supriyo Chakraborty et al. "Interpretability of deep learning models: a survey of results". In: *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE. 2017, pp. 1–6.

[91] André Altmann et al. "Permutation importance: a corrected feature importance measure". In: *Bioinformatics* 26.10 (2010), pp. 1340–1347.

[92] Dong Nguyen, Noah A Smith, and Carolyn Rose. "Author age prediction from text using linear regression". In: *Proceedings of the 5th ACL-HLT workshop on language technology for cultural heritage, social sciences, and humanities*. 2011, pp. 115–123.

[93] Shirish Krishnaj Shevade and S Sathiya Keerthi. "A simple and efficient algorithm for gene selection using sparse logistic regression". In: *Bioinformatics* 19.17 (2003), pp. 2246–2253.

[94] Lin Song, Peter Langfelder, and Steve Horvath. "Random generalized linear model: a highly accurate and interpretable ensemble predictor". In: *BMC bioinformatics* 14.1 (2013), p. 5.

[95] Kenji Kira and Larry A Rendell. "A practical approach to feature selection". In: *Machine Learning Proceedings 1992*. Elsevier, 1992, pp. 249–256.

[96] Tin Kam Ho. "The random subspace method for constructing decision forests". In: *IEEE transactions on pattern analysis and machine intelligence* 20.8 (1998), pp. 832–844.

[97] Koen W De Bock and Dirk Van den Poel. "Reconciling performance and interpretability in customer churn prediction using ensemble learning based on generalized additive models". In: *Expert Systems with Applications* 39.8 (2012), pp. 6816–6826.

[98] Binxu Zhai and Jianguo Chen. "Development of a stacked ensemble model for forecasting and analyzing daily average PM2.5 concentrations in Beijing, China". In: *Science of The Total Environment* 635 (2018), pp. 644–658. ISSN: 0048-9697.

[99] Robert Tibshirani. "Regression shrinkage and selection via the lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[100] Yoav Freund, Robert E Schapire, et al. “Experiments with a new boosting algorithm”. In: *icml*. Vol. 96. Citeseer. 1996, pp. 148–156.

[101] Gideon Schwarz et al. “Estimating the dimension of a model”. In: *The annals of statistics* 6.2 (1978), pp. 461–464.

[102] Carolin Strobl et al. “Bias in random forest variable importance measures: Illustrations, sources and a solution”. In: *BMC bioinformatics* 8.1 (2007), p. 25.

[103] Lloyd S Shapley. “A value for n-person games”. In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[104] Scott M. Lundberg et al. “From local explanations to global understanding with explainable AI for trees”. In: *Nature Machine Intelligence* 2.1 (2020), pp. 2522–5839.

[105] Ramprasaath R Selvaraju et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[106] Weili Nie, Yang Zhang, and Ankit Patel. “A theoretical explanation for perplexing behaviors of backpropagation-based visualizations”. In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3809–3818.

[107] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. “Learning important features through propagating activation differences”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3145–3153.

[108] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3319–3328.

[109] Julius Adebayo et al. "Sanity checks for saliency maps". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9505–9515.

[110] Marco Ancona et al. "Towards better understanding of gradient-based attribution methods for deep neural networks". In: *arXiv preprint arXiv:1711.06104* (2017).

[111] M. G. Kendall. *Rank correlation methods.* Oxford, England: Griffin, 1948.

[112] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[113] Bhekisipho Twala. "Impact of noise on credit risk prediction: Does data quality really matter?" In: *Intelligent Data Analysis* 17.6 (2013), pp. 1115–1134.

[114] Elias Kalapanidas et al. "Machine learning algorithms: a study on noise sensitivity". In: *Proc. 1st Balcan Conference in Informatics*. 2003, pp. 356–365.

[115] J Sola and Joaquin Sevilla. "Importance of input data normalization for the application of neural networks to complex industrial problems". In: *IEEE Transactions on nuclear science* 44.3 (1997), pp. 1464–1468.

[116] James Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization". In: *Journal of machine learning research* 13.Feb (2012), pp. 281–305.

[117] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. "Regularization for deep learning: A taxonomy". In: *arXiv preprint arXiv:1710.10686* (2017).

[118] Chris M. Bishop. "Training with Noise is Equivalent to Tikhonov Regularization". In: *Neural Computation* 7.1 (1995), pp. 108–116. DOI: 10.1162/neco.1995.7.1.108.

[119] Chee Kai Chua. *3D printing and additive manufacturing : principles and applications (the 5th edition of Rapid prototyping : principles and applications).* eng. 5th ed. / Chee Kai Chua, Kah Fai Leong. Singapore: World Scientific, 2017. ISBN: 9813146753.

[120] Michael F. Ashby and David R.H. Jones. "Chapter 1 - Engineering Materials and Their Properties". eng. In: *Engineering Materials 1.* Fourth Edition. Elsevier Ltd, 2012, pp. 1–12. ISBN: 9780080966656.

[121] Roger C Reed. *The superalloys [electronic resource] : fundamentals and applications / Roger C. Reed.* eng. Cambridge, UK ; New York: Cambridge University Press, 2006. ISBN: 1107167256.

[122] S. Babu et al. "Additive Manufacturing of Nickel Superalloys: Opportunities for Innovation and Challenges Related to Qualification". eng. In: *Metallurgical and Materials Transactions A* 49.9 (2018), pp. 3764–3780. ISSN: 1073-5623.

[123] Luke Nelson Carter. *Selective laser melting of nickel superalloys for high temperature applications.* eng. 2013.

[124] Salomé Sanchez et al. “Machine learning to determine the main factors affecting creep rates in laser powder bed fusion”. In: *Journal of Intelligent Manufacturing* 32.8 (2021), pp. 2353–2373.

[125] Connor Shorten and Taghi M Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of Big Data* 6.1 (2019), p. 60.

[126] Stefan Van Der Walt et al. “scikit-image: image processing in Python”. In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359.

[127] Cedric Seger. *An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing*. 2018.

[128] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[129] Xinbo Qi et al. “Applying neural-network-based machine learning to additive manufacturing: current applications, challenges, and future perspectives”. In: *Engineering* 5.4 (2019), pp. 721–729.

[130] Bin Zhang, Shunyu Liu, and Yung C. Shin. “In-Process monitoring of porosity during laser additive manufacturing process”. In: *Additive Manufacturing* 28 (2019), pp. 497–505. ISSN: 2214-8604.

[131] Jack Francis and Linkan Bian. “Deep Learning for Distortion Prediction in Laser-Based Additive Manufacturing using Big Data”. In: *Manufacturing Letters* 20 (2019), pp. 10–14. ISSN: 2213-8463.

[132] Lijun Song et al. "Real-time composition monitoring using support vector regression of laser-induced plasma for laser additive manufacturing". In: *IEEE Transactions on Industrial Electronics* 64.1 (2016), pp. 633–642.

[133] Zhixiong Li et al. "Prediction of surface roughness in extrusion-based additive manufacturing with machine learning". In: *Robotics and Computer-Integrated Manufacturing* 57 (2019), pp. 488–495.

[134] F. Pedregosa et al. "Scikit-learn: machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[135] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[136] Divish Rengasamy et al. "An Intelligent Toolkit for Benchmarking Data-Driven Aerospace Prognostics". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 4210–4215.

[137] Andres Bustillo et al. "Improving the accuracy of machine-learning models with data from machine test repetitions". In: *Journal of Intelligent Manufacturing* (2020), pp. 1–19.

[138] Jimiama Mafeni Mase et al. "Capturing Uncertainty in Heavy Goods Vehicles Driving Behaviour". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2020, pp. 1–7.

[139] Lotfi A Zadeh. "Fuzzy sets". In: *Information and control* 8.3 (1965), pp. 338–353.

[140] L-X Wang and Jerry M Mendel. "Generating fuzzy rules by learning from examples". In: *IEEE Trans. Syst., Man, Cybern.* 22.6 (1992), pp. 1414–1427.

[141] Ebrahim H Mamdani and Sedrak Assilian. "An experiment in linguistic synthesis with a fuzzy logic controller". In: *International journal of man-machine studies* 7.1 (1975), pp. 1–13.

[142] L-X Wang and Jerry M Mendel. "Generating fuzzy rules by learning from examples". In: *IEEE Transactions on systems, man, and cybernetics* 22.6 (1992), pp. 1414–1427.

[143] Daniel Berrar. "Cross-validation". In: *Encyclopedia of bioinformatics and computational biology* 1 (2019), pp. 542–545.

[144] Jerome H Friedman, Bogdan E Popescu, et al. "Predictive learning via rule ensembles". In: *Annals of Applied Statistics* 2.3 (2008), pp. 916–954.

[145] David A Winkler and Tu C Le. "Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and QSAR". In: *Molecular informatics* 36.1-2 (2017), p. 1600118.

[146] Zhengkai Xu et al. "Defect evolution in laser powder bed fusion additive manufactured components during thermo-mechanical testing". In: 2017.

[147] Luke Sheridan et al. "Relating porosity to fatigue failure in additively manufactured alloy 718". In: *Materials Science and Engineering: A* 727 (2018), pp. 170–176. ISSN: 0921-5093.

[148] Zk Xu et al. “Staged thermomechanical testing of nickel superalloys produced by selective laser melting”. English. In: *Materials & Design* 133 (2017), pp. 520–527. ISSN: 0264-1275.

[149] A.M. Mancisidor et al. “Reduction of the Residual Porosity in Parts Manufactured by Selective Laser Melting Using Skywriting and High Focus Offset Strategies”. In: *Physics Procedia* 83 (2016). Laser Assisted Net Shape Engineering 9 International Conference on Photonic Technologies Proceedings of the LANE 2016 September 19-22, 2016 Fürth, Germany, pp. 864–873. ISSN: 1875-3892.

[150] S. Sanchez et al. “The creep behaviour of nickel alloy 718 manufactured by laser powder bed fusion”. In: *Materials  Design* 204 (2021), p. 109647. ISSN: 0264-1275.

# Appendix A

# Appendix

## A.1 Supplementary results for Multi-Method Ensemble

### A.1.1 Feature Importance Quantification on Train and Test Dataset (RMSE)

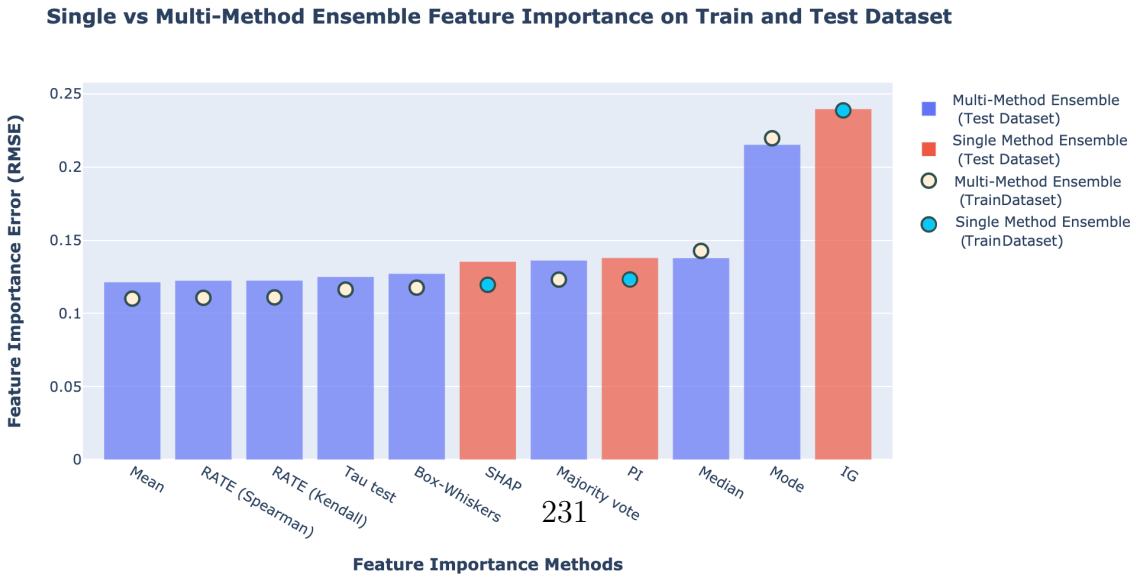



Figure A.1: Average feature importance error between SME and MME with train and test dataset. (RMSE)

### A.1.2 Feature Importance Quantification on Train and Test Dataset ($R^2$)

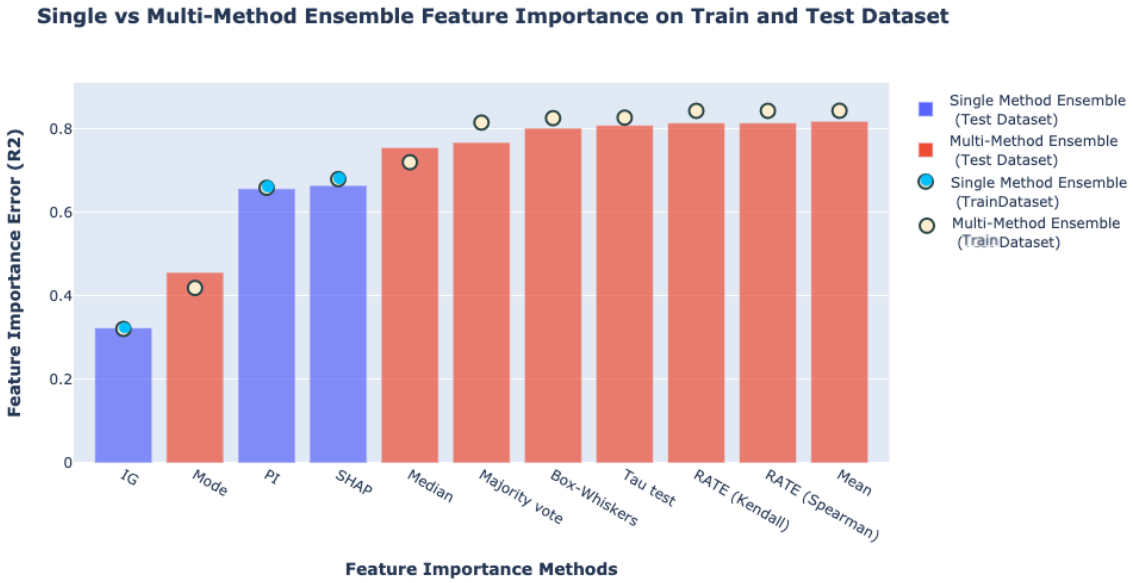



Figure A.2: Average feature importance error between SME and MME with train and test dataset. ($R^2$)

### A.1.3 Effect of Noise Level on All Feature Importance (RMSE)

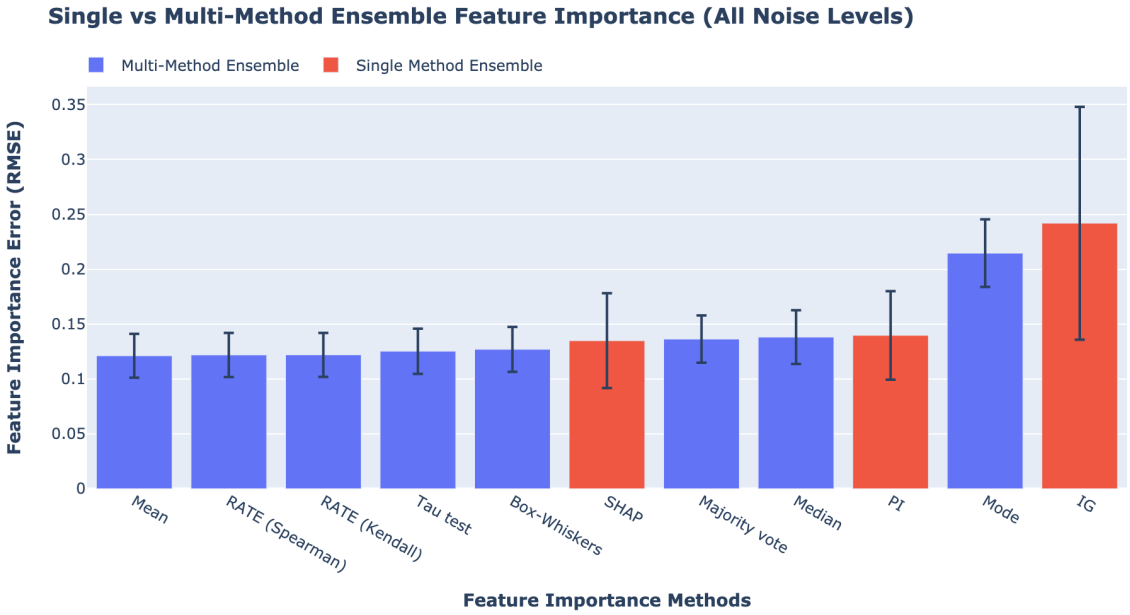



Figure A.3: Effect of all noise level on all feature importance methods (RMSE)

| | Models | Noise level (Standard deviation) 0 ($10^{-2}$) | 2 ($10^{-2}$) | 4 ($10^{-2}$) |
|---|---|---|---|---|
| SME | PI | 13.8±2.1 | 13.4±2.0 | 14.5±2.7 |
| | SHAP | 13.4±2.4 | 13.3±2.4 | 13.6±2.5 |
| | IG | 23.1±10.6 | 25.0±10.5 | 24.3±10.6 |
| MME | RATE (Kendall) | 12.0±3.2 | 11.9±3.3 | 12.6±3.7 |
| | RATE (Spearman) | 12.0±3.3 | 11.9±3.3 | 12.5±3.7 |
| | Median | 13.8±4.1 | 13.8±3.2 | 14.3±4.6 |
| | Mean | 12.0±3.2 | 11.8±3.3 | 12.5±3.7 |
| | Mode | 22.8±5.4 | 19.9±5.3 | 21.5±5.1 |
| | Box-whiskers | 12.5±3.4 | 12.6±3.4 | 12.9±3.7 |
| | Tau test | 12.3±3.4 | 12.2±3.3 | 13.0±3.8 |
| | Majority vote | 13.3±3.7 | 13.7±3.4 | 13.7±3.9 |

Table A.1: Summary of feature importance RMSE between different SME and MME for different noise level.

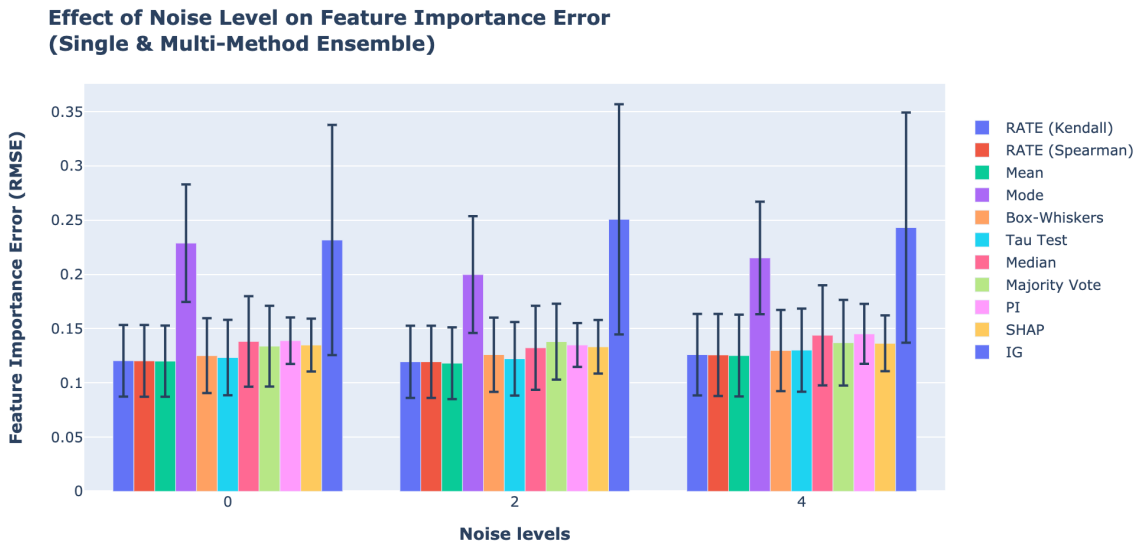



Figure A.4: Effect of noise levels on ensemble feature importance. (RMSE)

### A.1.4 Effect of Noise Level on All Feature Importance ($R^2$)

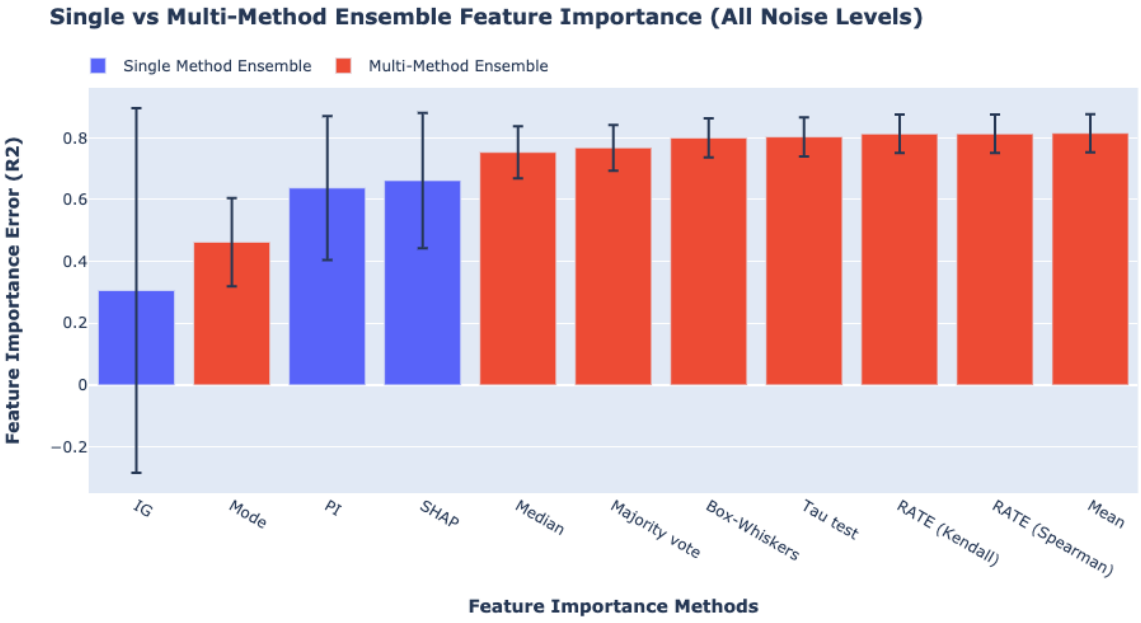



Figure A.5: Effect of all noise level on all feature importance methods ($R^2$)

| | | Noise level (Standard deviation) | | |
|---|---|---|---|---|
| | Models | 0 | 2 | 4 |
| 0. SME | PI | 0.63±0.11 | 0.67±0.10 | 0.60±0.17 |
| | SHAP | 0.65±0.12 | 0.67±0.11 | 0.65±0.13 |
| | IG | 0.29±0.59 | 0.28±0.59 | 0.34±0.59 |
| MME | RATE (Kendall) | 0.81±0.09 | 0.82±0.10 | 0.79±0.11 |
| | RATE (Spearman) | 0.81±0.09 | 0.82±0.10 | 0.79±0.11 |
| | Median | 0.75±0.15 | 0.78±0.12 | 0.72±0.15 |
| | Mean | 0.81±0.09 | 0.82±0.10 | 0.79±0.11 |
| | Mode | 0.39±0.26 | 0.52±0.25 | 0.46±0.22 |
| | Box-whiskers | 0.80±0.10 | 0.80±0.10 | 0.79±0.11 |
| | Tau test | 0.80±0.11 | 0.81±0.10 | 0.78±0.11 |
| | Majority vote | 0.77±0.12 | 0.76±0.12 | 0.76±0.13 |

Table A.2: Summary of feature importance $R^2$ between different SME and MME for different noise level.

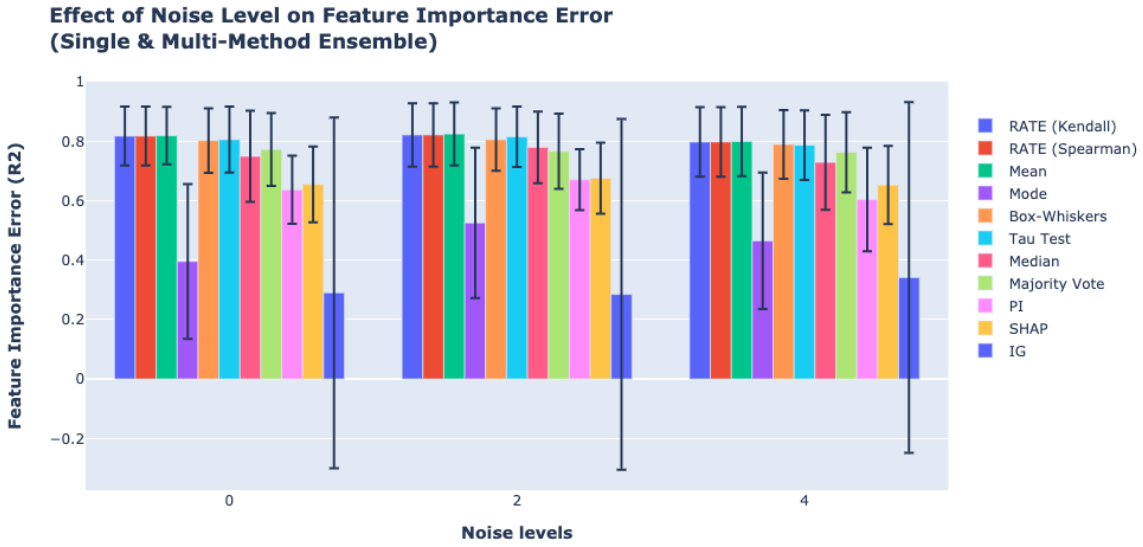



Figure A.6: Effect of noise levels on ensemble feature importance. ($R^2$)

### A.1.5 Effect Informative Level on All Feature Importance (RMSE)

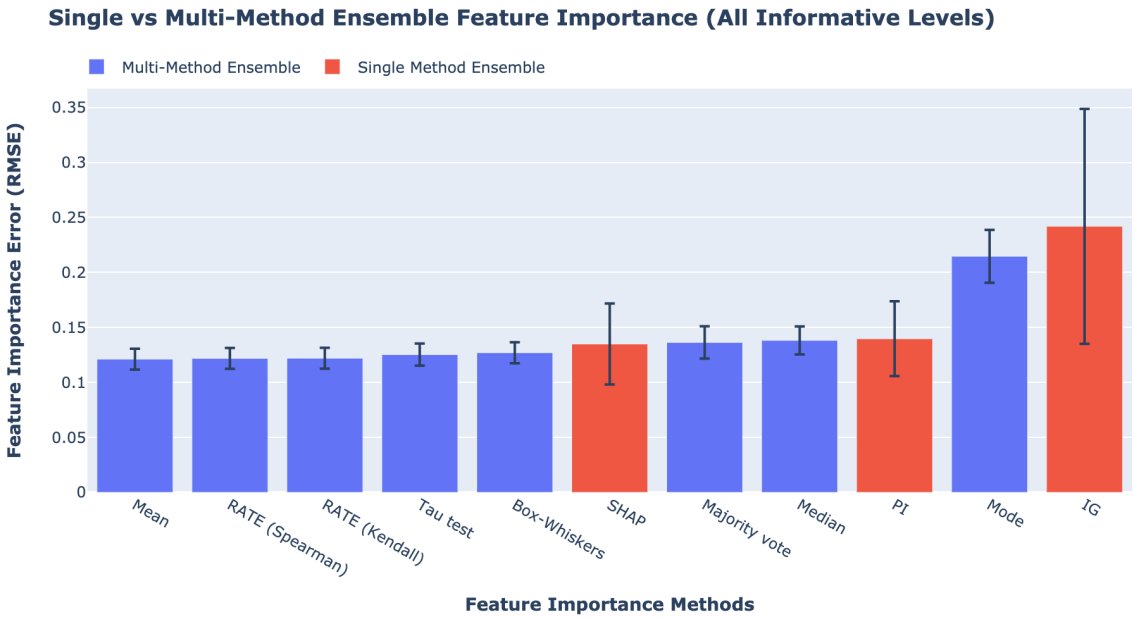



Figure A.7: Effect of feature informative level on all ensemble feature importance methods. (RMSE)

| | Models | Features informative level (%) | | | | |
|---|---|---|---|---|---|---|
| | | 20 ($10^{-2}$) | 40 ($10^{-2}$) | 60 ($10^{-2}$) | 80 ($10^{-2}$) | 100 ($10^{-2}$) |
| SME | PI | 6.0±0.9 | 11.1±1.2 | 15.6±2.3 | 17.4±1.2 | 19.5±1.4 |
| | SHAP | 5.1±0.8 | 9.9±1.3 | 14.0±1.5 | 18.0±1.9 | 20.2±2.1 |
| | IG | 14.9±10.6 | 19.7±10.6 | 24.0±10.6 | 26.7±10.6 | 32.1±10.6 |
| MME | RATE (Kendall) | 5.0±1.1 | 9.2±1.7 | 13.2±2.1 | 15.3±1.9 | 18.1±3.1 |
| | RATE (Spearman) | 5.0±1.1 | 9.2±1.7 | 13.2±2.1 | 15.3±1.9 | 18.1±3.1 |
| | Median | 5.1±1.5 | 10.8±2.5 | 15.3±2.7 | 17.5±3.1 | 20.1±3.7 |
| | Mean | 4.9±1.4 | 9.2±1.7 | 13.1±2.0 | 15.2±1.9 | 17.9±3.1 |
| | Mode | 20.2±4.3 | 18.8±6.0 | 22.9±6.1 | 22.6±4.8 | 22.5±5.3 |
| | Box-whiskers | 5.1±1.3 | 9.8±1.8 | 13.7±2.0 | 16.2±2.0 | 18.5±3.1 |
| | Tau test | 5.0±1.4 | 9.7±1.8 | 13.7±2.2 | 15.8±2.0 | 18.2±3.1 |
| | Majority vote | 8.0±2.4 | 12.2±3.1 | 15.3±2.8 | 15.1±3.2 | 16.9±4.4 |

Table A.3: Summary of feature importance RMSE between different SME and MME for different percentage of informative level.

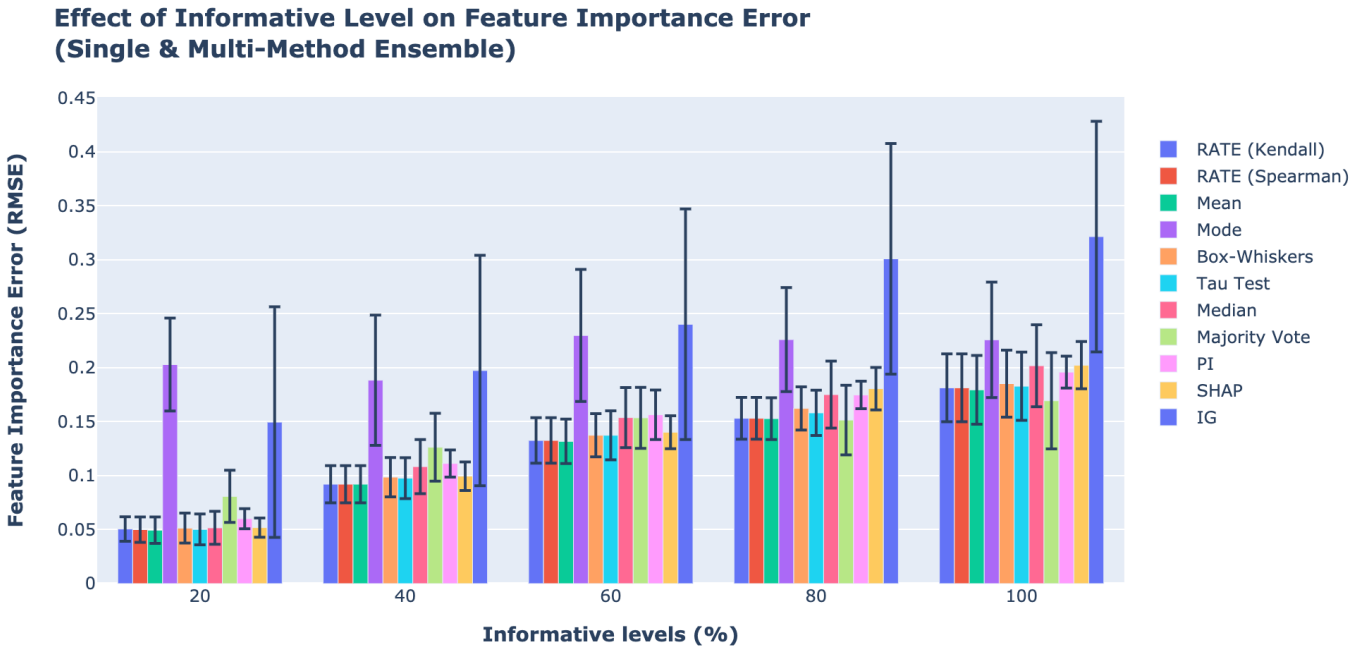



Figure A.8: Effect of feature informative levels on ensemble feature importance methods. (RMSE)

### A.1.6 Effect Informative Level on All Feature Importance ($R^2$)

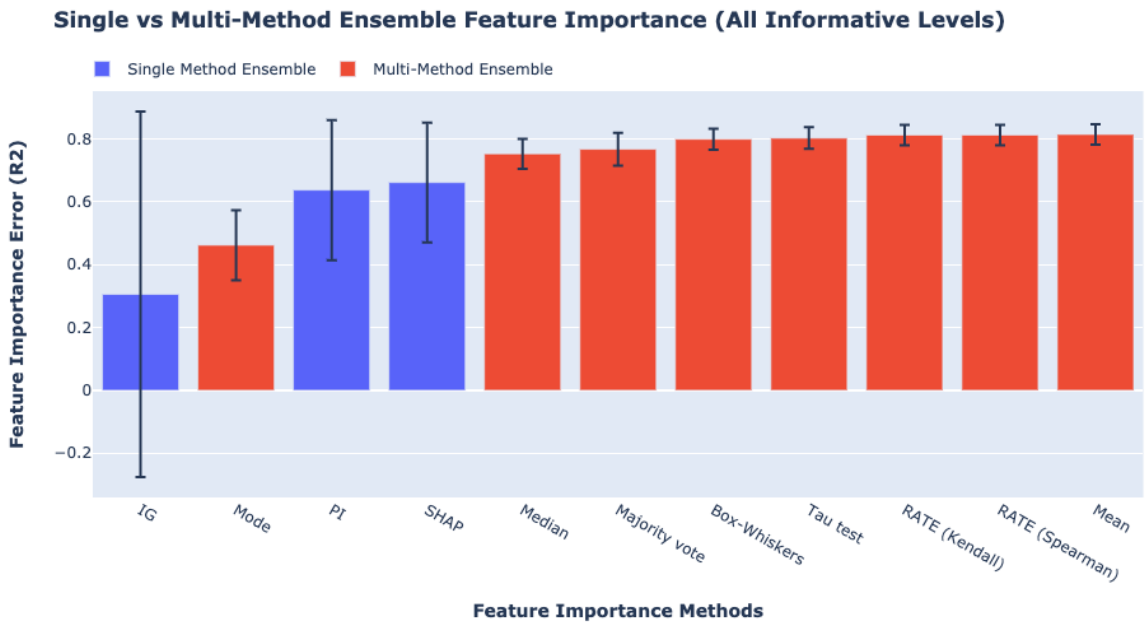



Figure A.9: Effect of feature informative level on all ensemble feature importance methods. ($R^2$)

| | Models | Features informative level (%) | | | | |
|---|---|---|---|---|---|---|
| | | 20 | 40 | 60 | 80 | 100 |
| SME | PI | 0.90±0.02 | 0.79±0.04 | 0.63±0.16 | 0.57±0.06 | 0.27±0.12 |
| | SHAP | 0.92±0.02 | 0.83±0.04 | 0.73±0.05 | 0.55±0.08 | 0.25±0.15 |
| | IG | 0.65±0.58 | 0.58±0.58 | 0.44±0.58 | 0.17±0.58 | -0.32±0.58 |
| MME | RATE (Kendall) | 0.95±0.01 | 0.90±0.03 | 0.83±0.05 | 0.78±0.05 | 0.57±0.13 |
| | RATE (Spearman) | 0.95±0.01 | 0.90±0.03 | 0.83±0.05 | 0.78±0.05 | 0.57±0.13 |
| | Median | 0.95±0.02 | 0.86±0.05 | 0.77±0.07 | 0.70±0.09 | 0.46±0.19 |
| | Mean | 0.95±0.01 | 0.90±0.03 | 0.83±0.05 | 0.78±0.05 | 0.58±0.13 |
| | Mode | 0.37±0.22 | 0.57±0.28 | 0.49±0.24 | 0.52±0.19 | 0.34±0.29 |
| | Box-whiskers | 0.95±0.02 | 0.89±0.03 | 0.82±0.05 | 0.75±0.06 | 0.56±0.13 |
| | Tau test | 0.95±0.02 | 0.89±0.03 | 0.82±0.05 | 0.76±0.06 | 0.57±0.14 |
| | Majority vote | 0.88±0.05 | 0.81±0.07 | 0.77±0.08 | 0.77±0.09 | 0.59±0.20 |

Table A.4: Summary of feature importance $R^2$ between different SME and MME for different percentage of informative level.

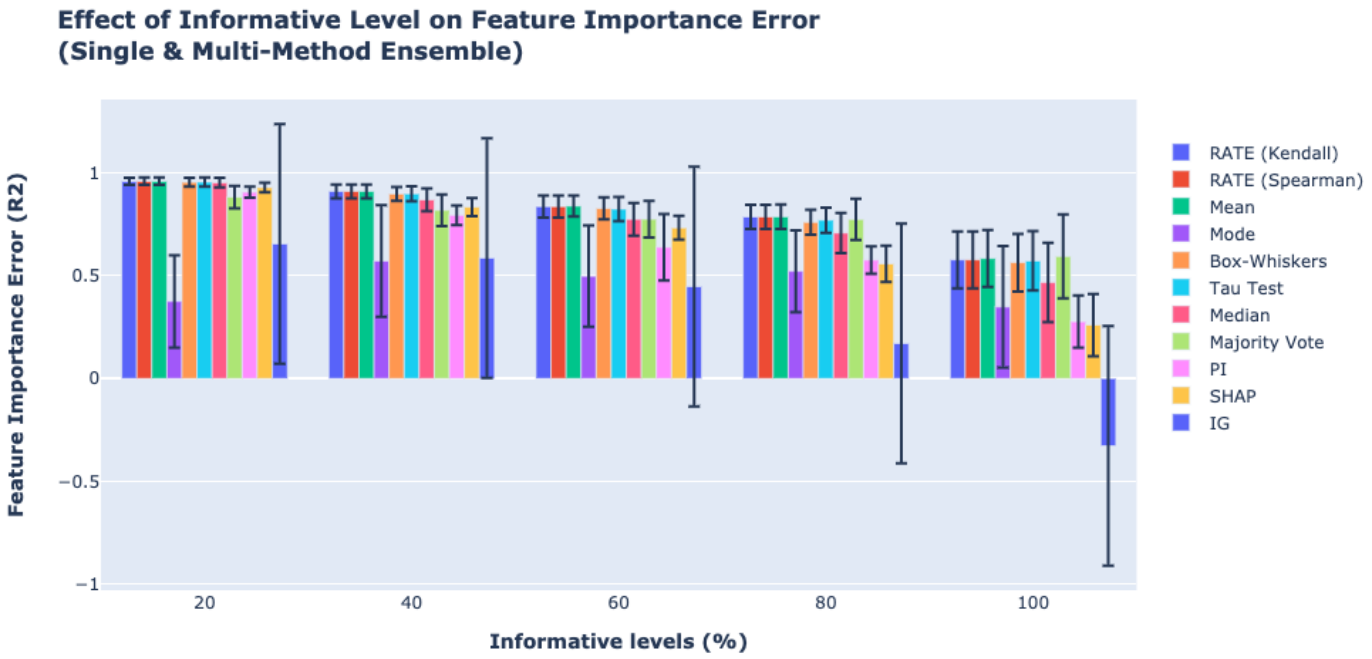



Figure A.10: Effect of feature informative levels on ensemble feature importance methods. ($R^2$)

### A.1.7 Effect of Number of Features on All Feature Importance (RMSE)

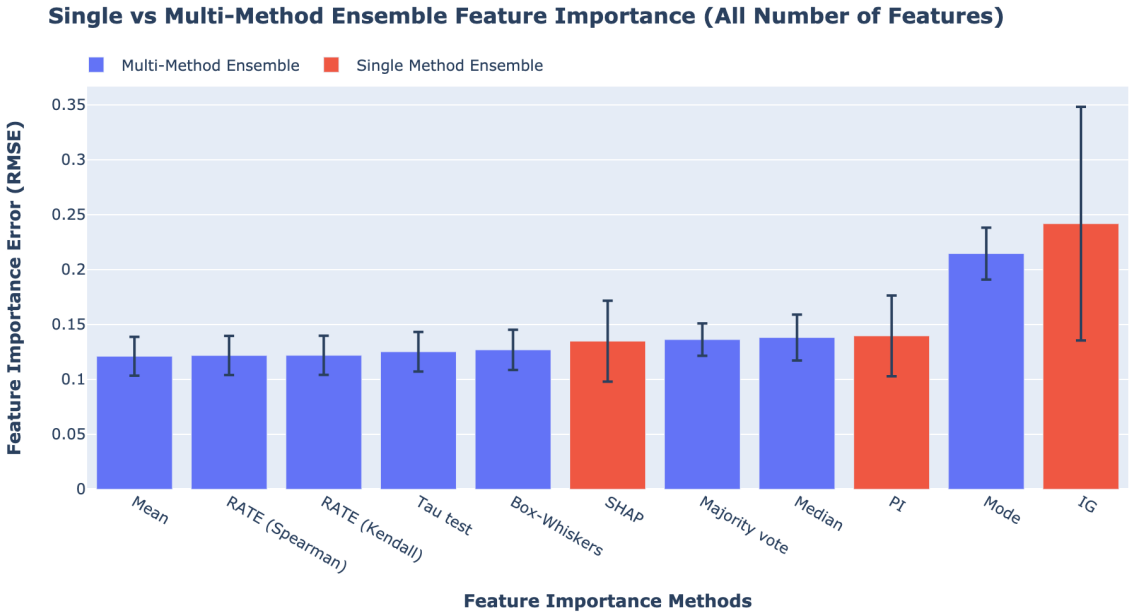



Figure A.11: Effect of number of features on all ensemble feature importance methods. (RMSE)

| | Models | Number of features | | |
|---|---|---|---|---|
| | | 20 ($10^{-2}$) | 60 ($10^{-2}$) | 100 ($10^{-2}$) |
| SME | PI | 10.3±1.9 | 14.9±1.9 | 16.5±2.4 |
| | SHAP | 8.5±1.6 | 14.2±2.2 | 17.6±2.4 |
| | IG | 23.6±10.6 | 24.0±10.6 | 24.7±10.6 |
| MME | RATE (Kendall) | 8.6±2.5 | 12.7±3.0 | 15.1±3.6 |
| | RATE (Spearman) | 8.67±2.5 | 12.7±3.0 | 15.1±3.6 |
| | Median | 8.6±2.8 | 15.4±3.9 | 17.2±4.0 |
| | Mean | 8.5±2.5 | 12.7±2.9 | 15.0±3.6 |
| | Mode | 29.7±5.8 | 18.3±3.3 | 16.2±2.2 |
| | Box-whiskers | 9.0±2.8 | 13.5±3.1 | 15.4±3.5 |
| | Tau test | 8.5±2.7 | 13.4±3.0 | 15.5±3.5 |
| | Majority vote | 8.3±2.3 | 13.0±3.1 | 19.4±3.3 |

Table A.5: Summary of feature importance RMSE between different SME and MME for different number of features.

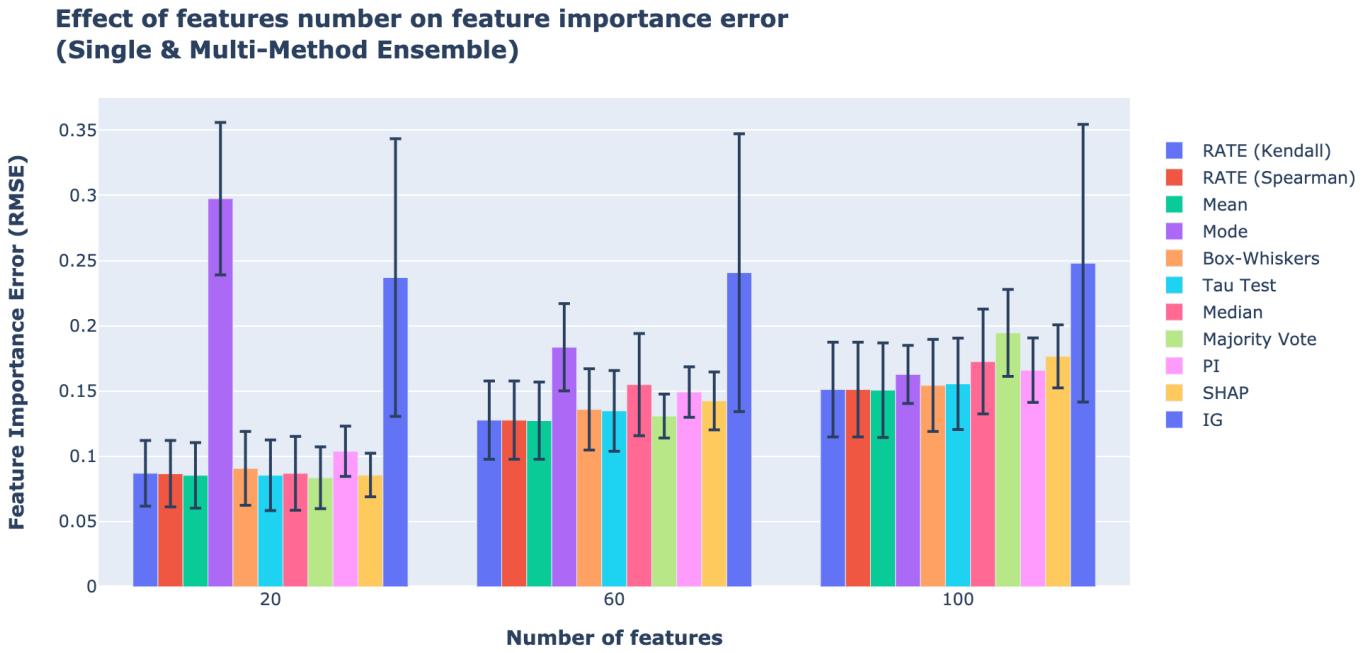



Figure A.12: Effect of number of features on ensemble feature importance methods. (RMSE)

### A.1.8 Effect of Number of Features on All Feature Importance ($R^2$)

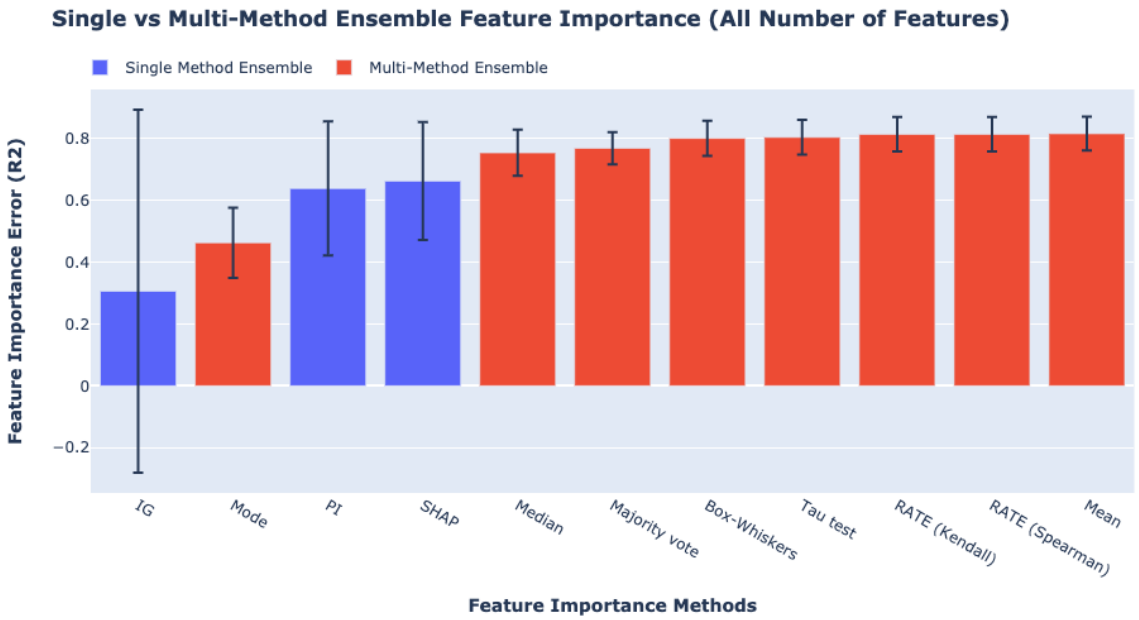



Figure A.13: Effect of number of features on all ensemble feature importance methods. ($R^2$)

| | | Number of features | | |
|---|---|---|---|---|
| | Models | 20 | 60 | 100 |
| SME | PI | 0.81±0.07 | 0.60±0.11 | 0.50±0.16 |
| | SHAP | 0.87±0.05 | 0.62±0.11 | 0.48±0.13 |
| | IG | 0.34±0.58 | 0.32±0.58 | 0.24±0.58 |
| MME | RATE (Kendall) | 0.91±0.04 | 0.80±0.09 | 0.71±0.12 |
| | RATE (Spearman) | 0.91±0.04 | 0.80±0.09 | 0.71±0.29 |
| | Median | 0.91±0.04 | 0.71±0.14 | 0.63±0.16 |
| | Mean | 0.91±0.04 | 0.80±0.09 | 0.71±0.12 |
| | Mode | 0.09±0.29 | 0.60±0.14 | 0.68±0.08 |
| | Box-whiskers | 0.90±0.04 | 0.78±0.09 | 0.70±0.13 |
| | Tau test | 0.91±0.04 | 0.78±0.09 | 0.70±0.13 |
| | Majority vote | 0.92±0.03 | 0.81±0.04 | 0.56±0.14 |

Table A.6: Summary of feature importance $R^2$ between different SME and MME for different number of features.

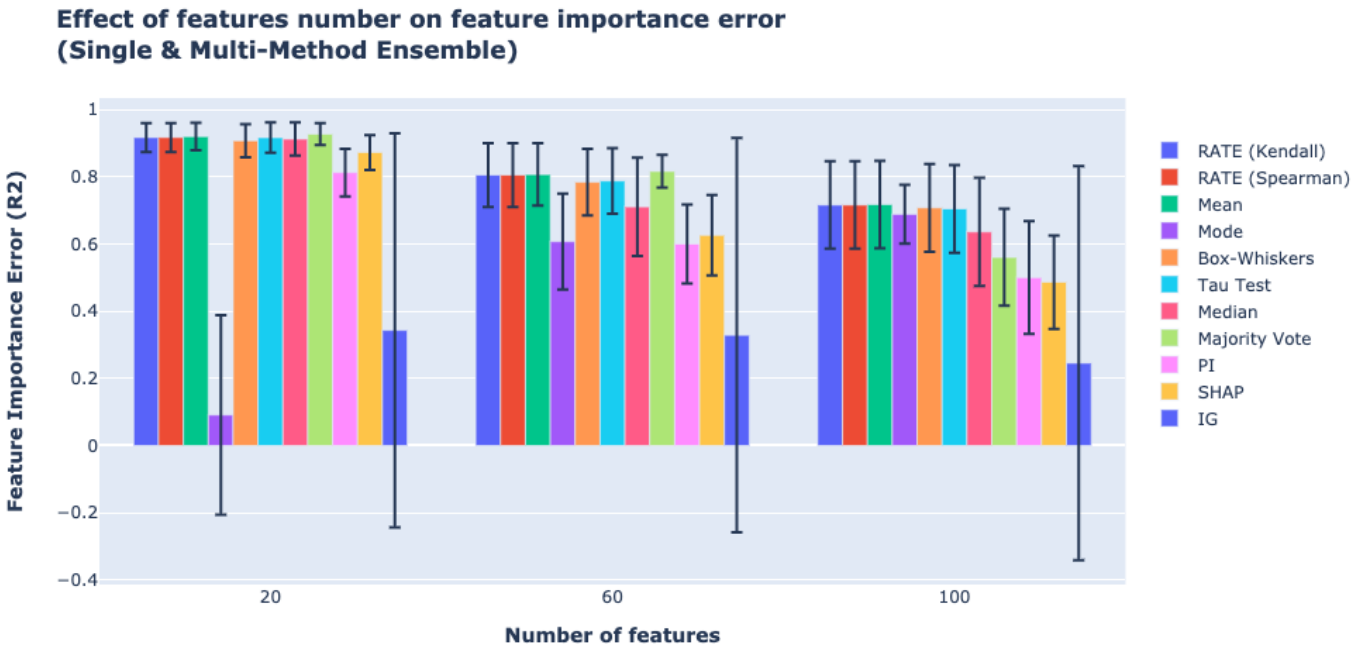



Figure A.14: Effect of number of features on ensemble feature importance methods. ($R^2$)

## A.2 Supplementary results for Fuzzy Ensemble Feature Importance

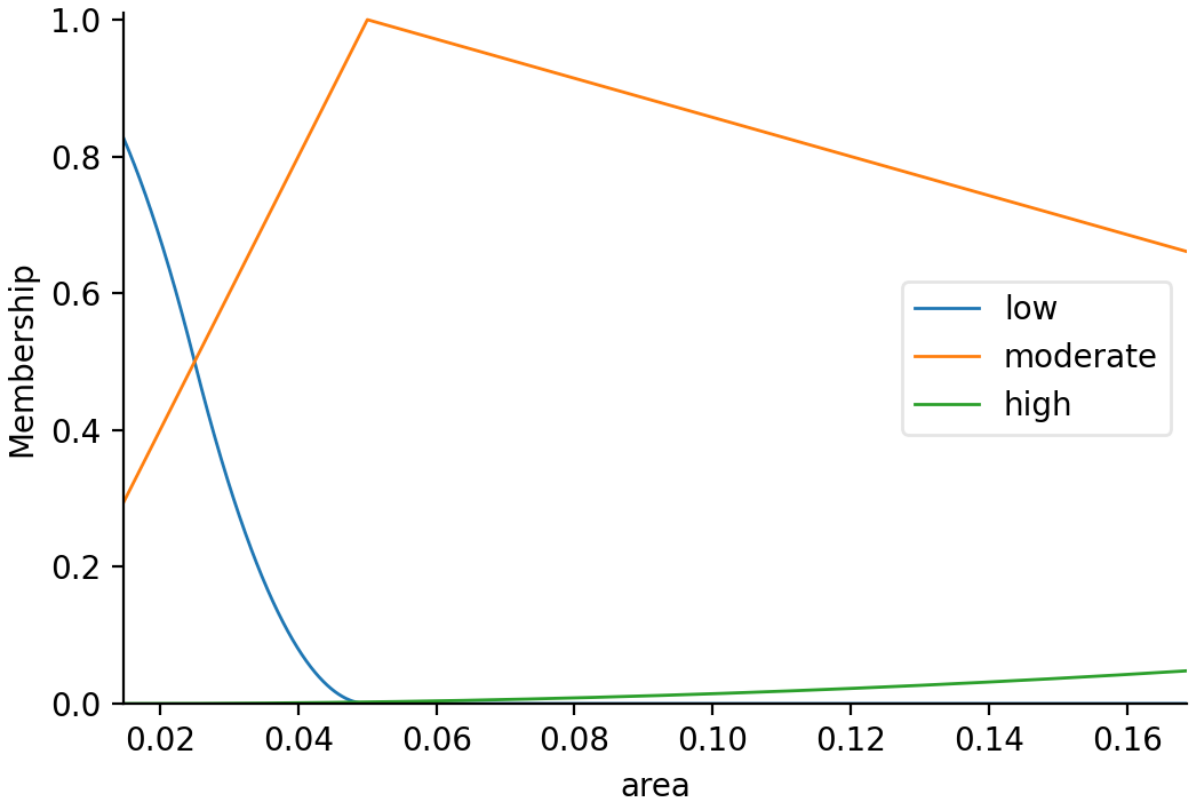



Figure A.15: Fuzzy membership of material descriptor: Area in creep rate prediction.

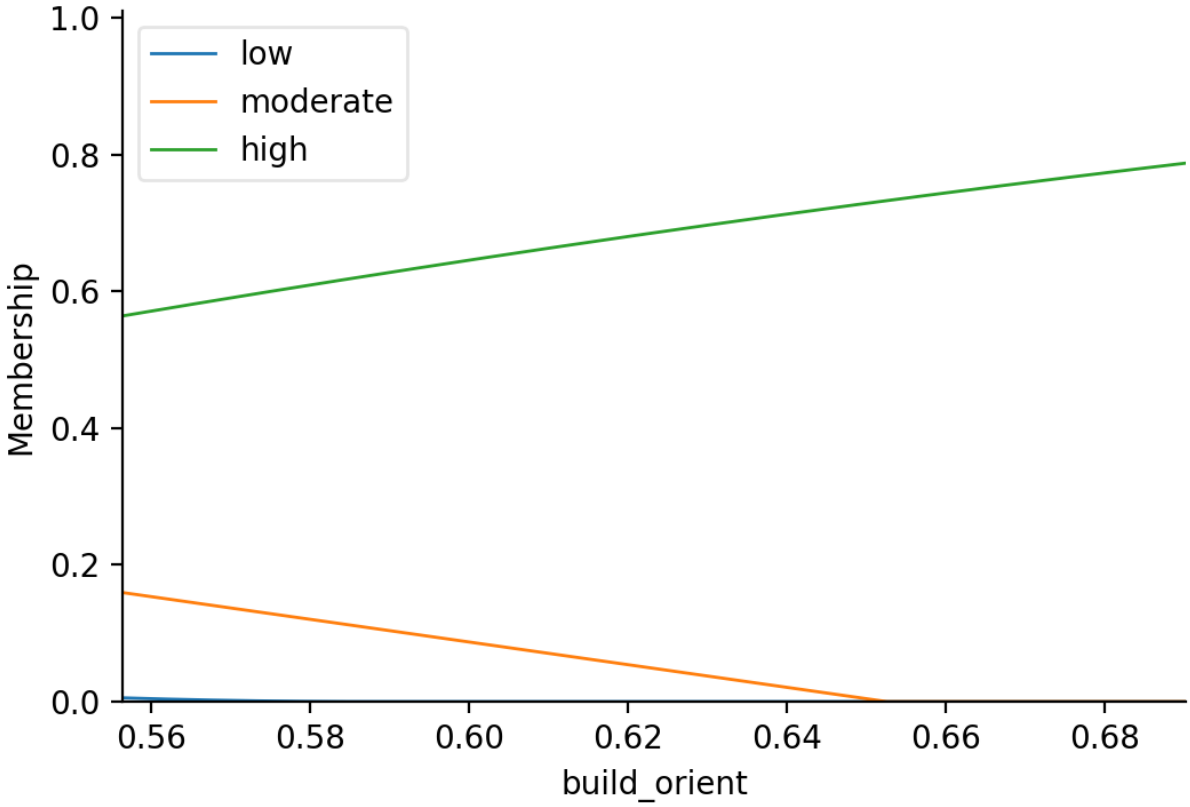



Figure A.16: Fuzzy membership of material descriptor: Build Orientation in creep rate prediction.

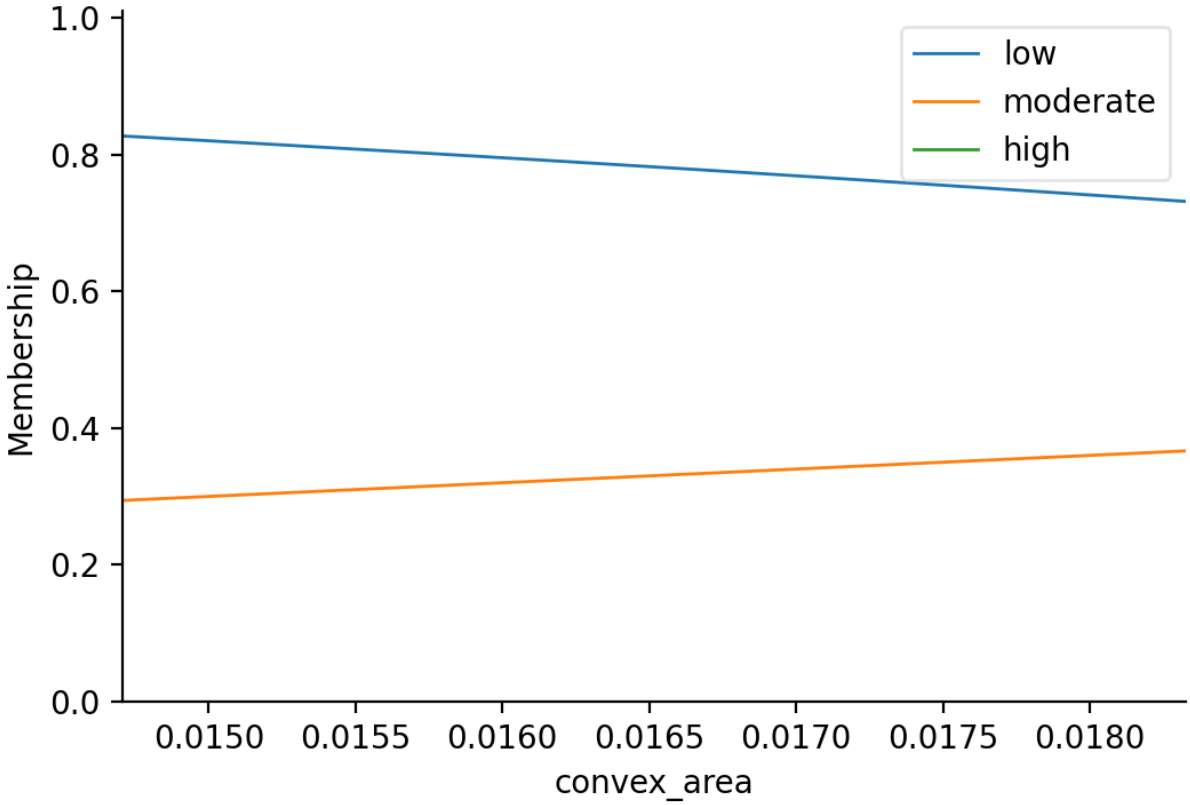



Figure A.17: Fuzzy membership of material descriptor: convex area in creep rate prediction.

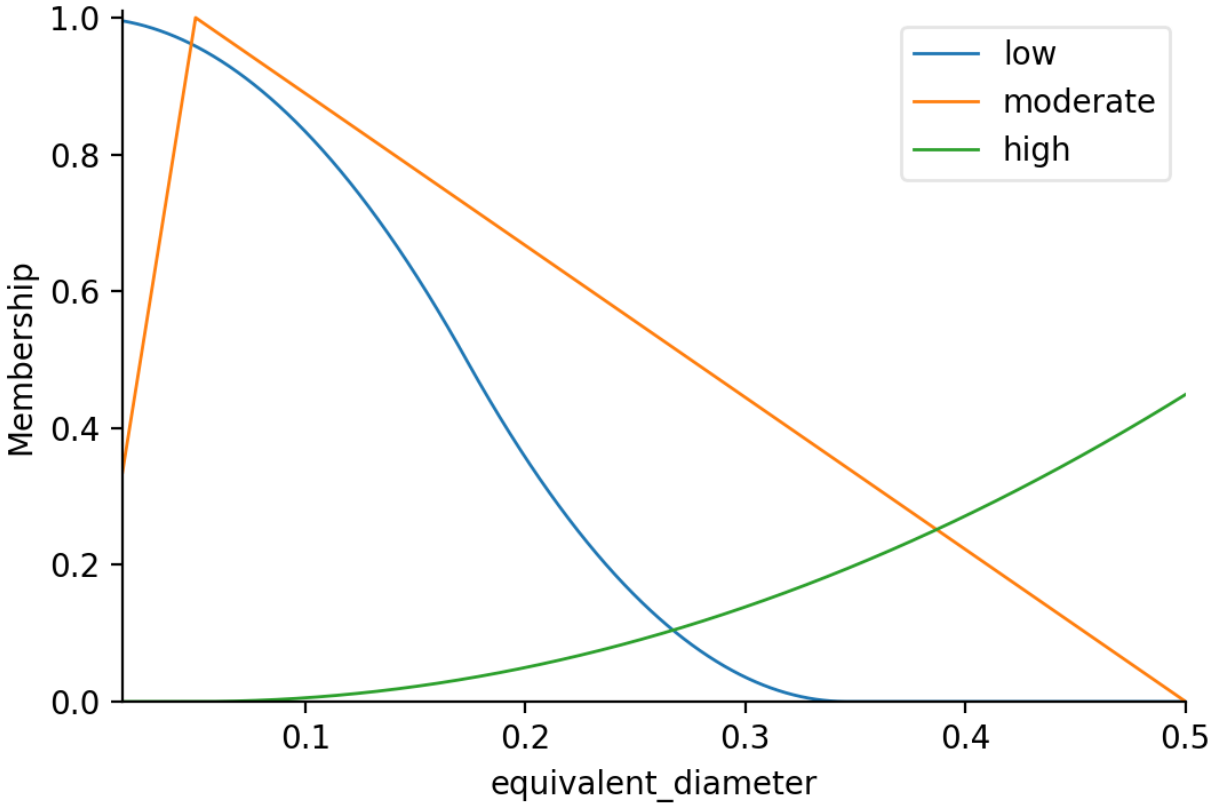



Figure A.18: Fuzzy membership of material descriptor: equivalent diameter in creep rate prediction.

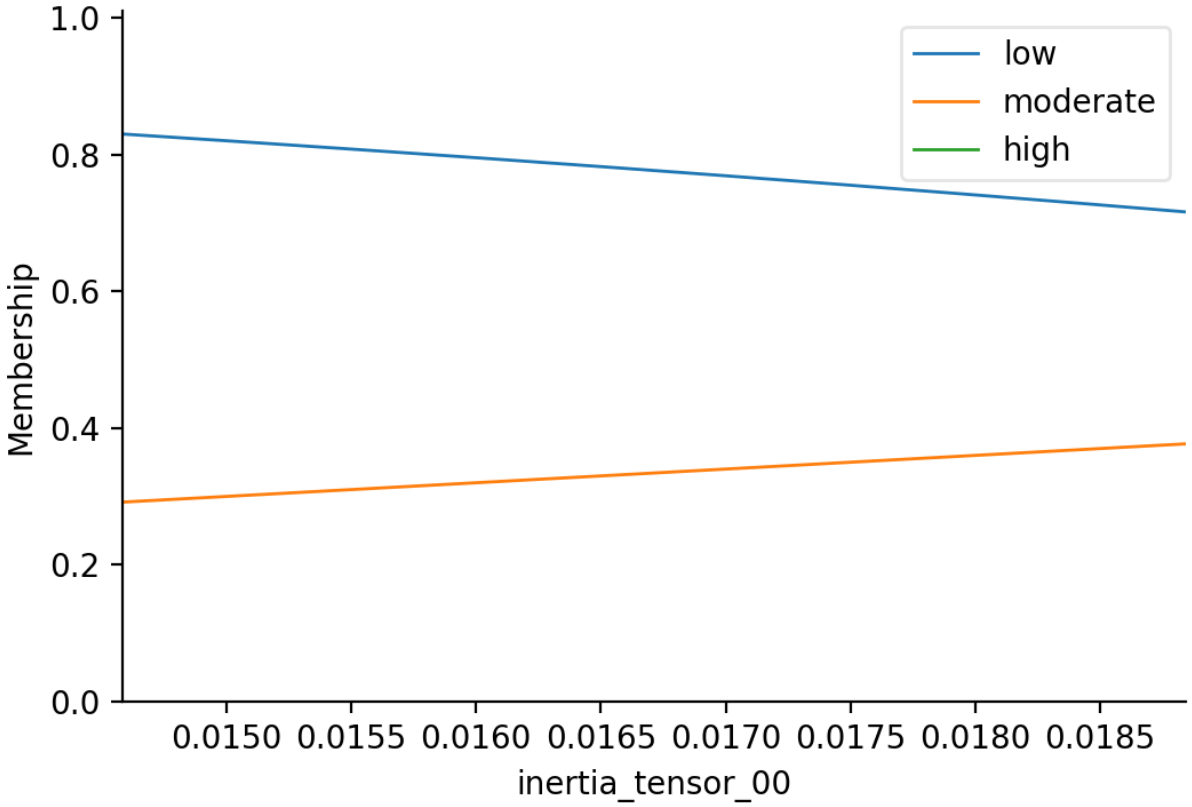



Figure A.19: Fuzzy membership of material descriptor: Inertia Tensor 00 in creep rate prediction.

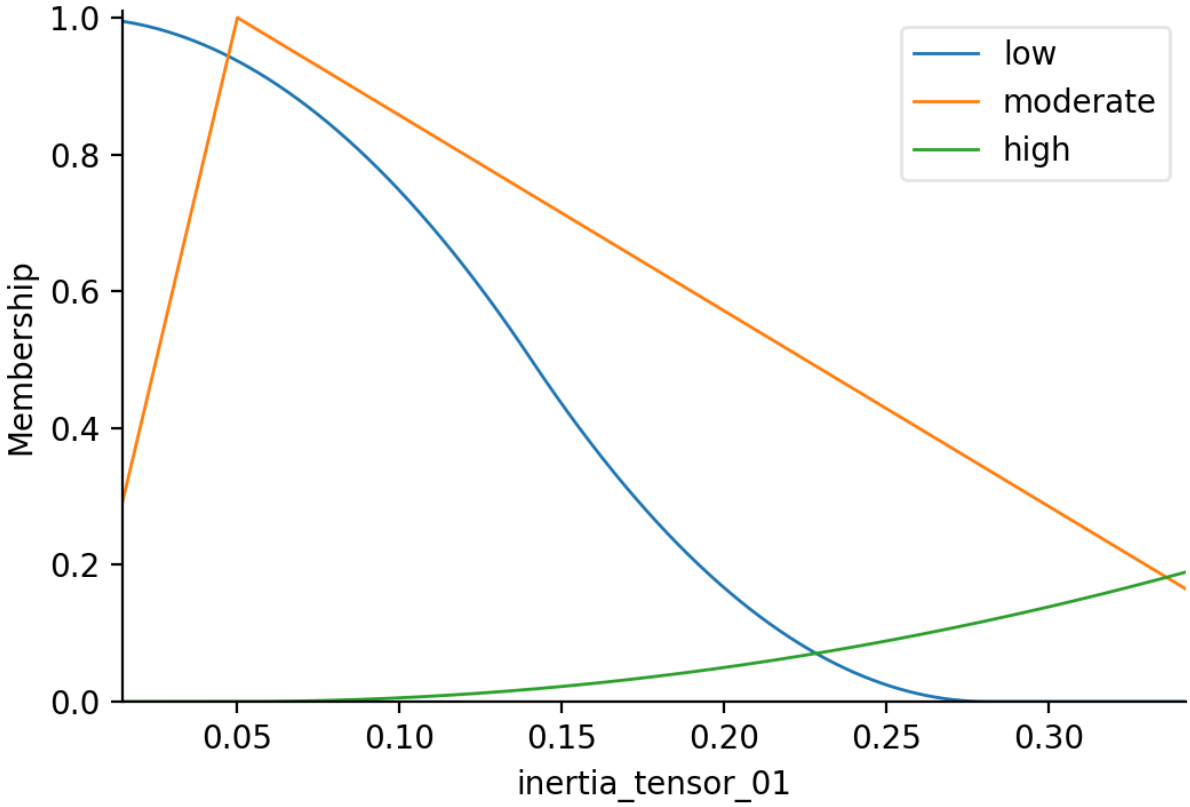



Figure A.20: Fuzzy membership of material descriptor: Inertia Tensor 01 in creep rate prediction.

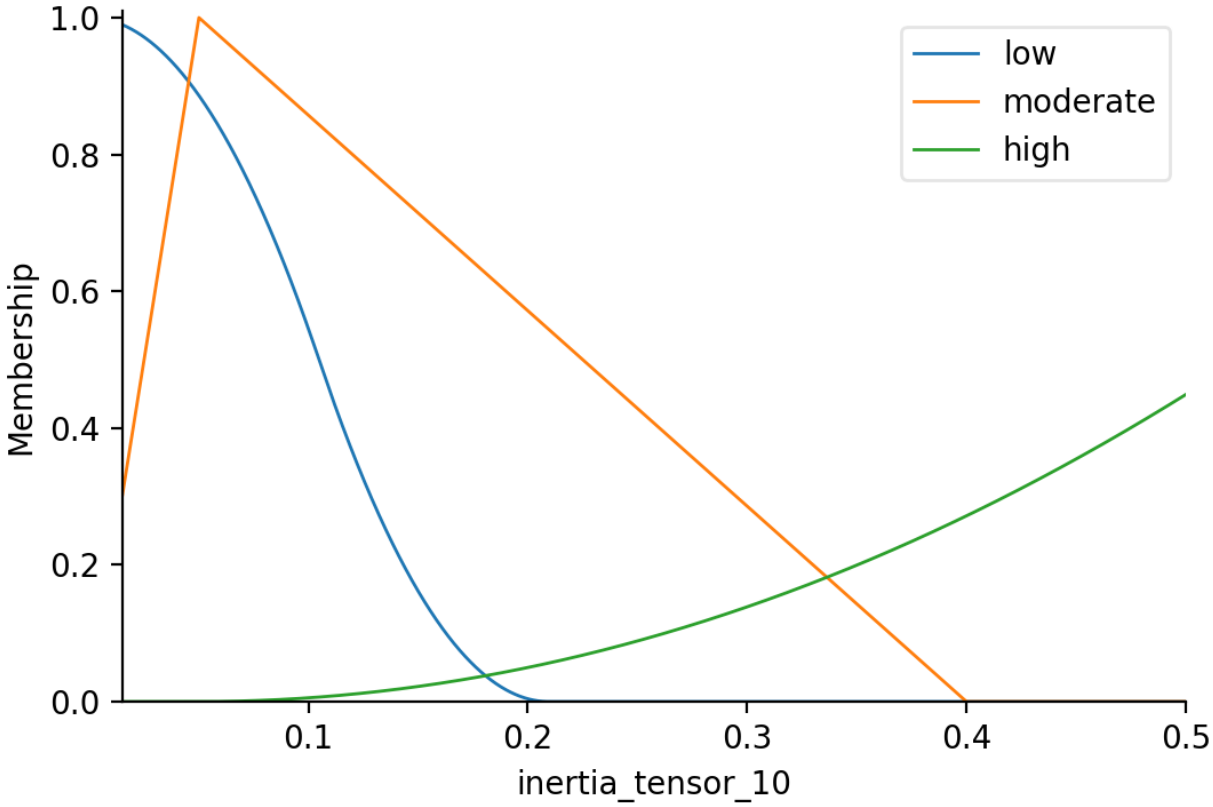



Figure A.21: Fuzzy membership of material descriptor: Inertia Tensor 10 in creep rate prediction.

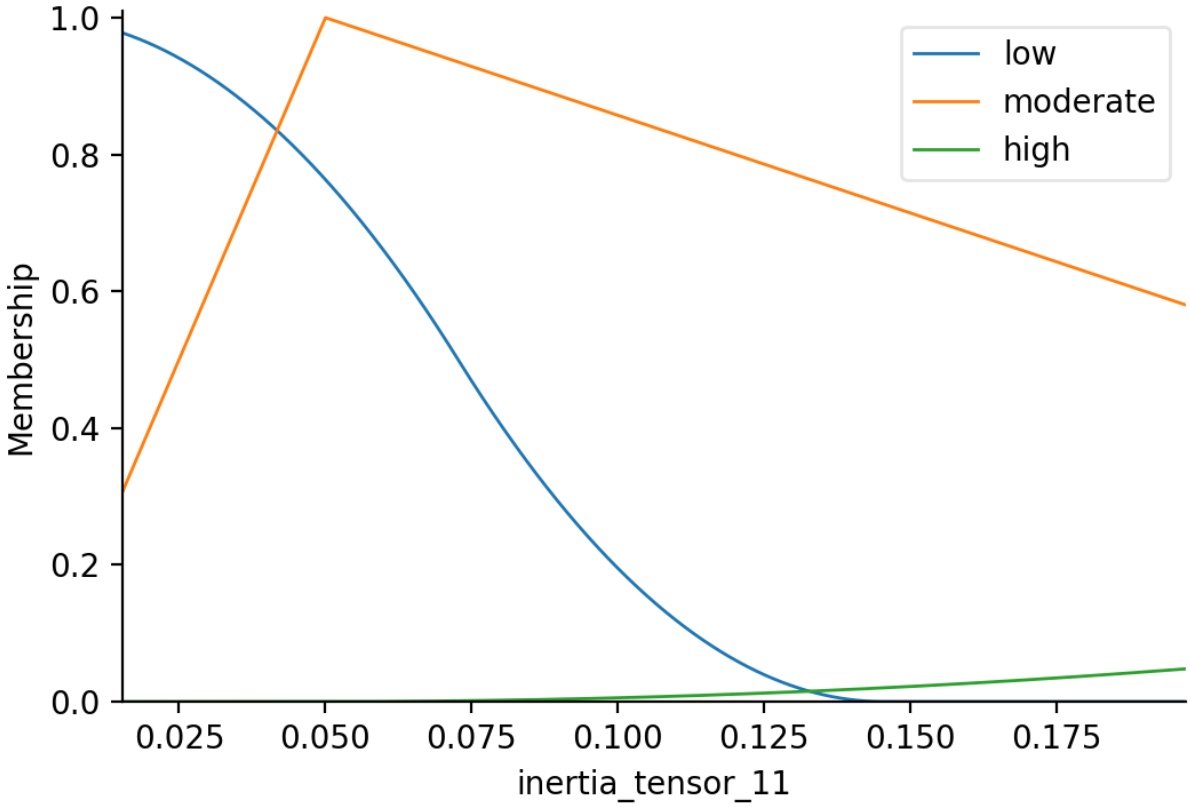



Figure A.22: Fuzzy membership of material descriptor: Inertia Tensor 11 in creep rate prediction.

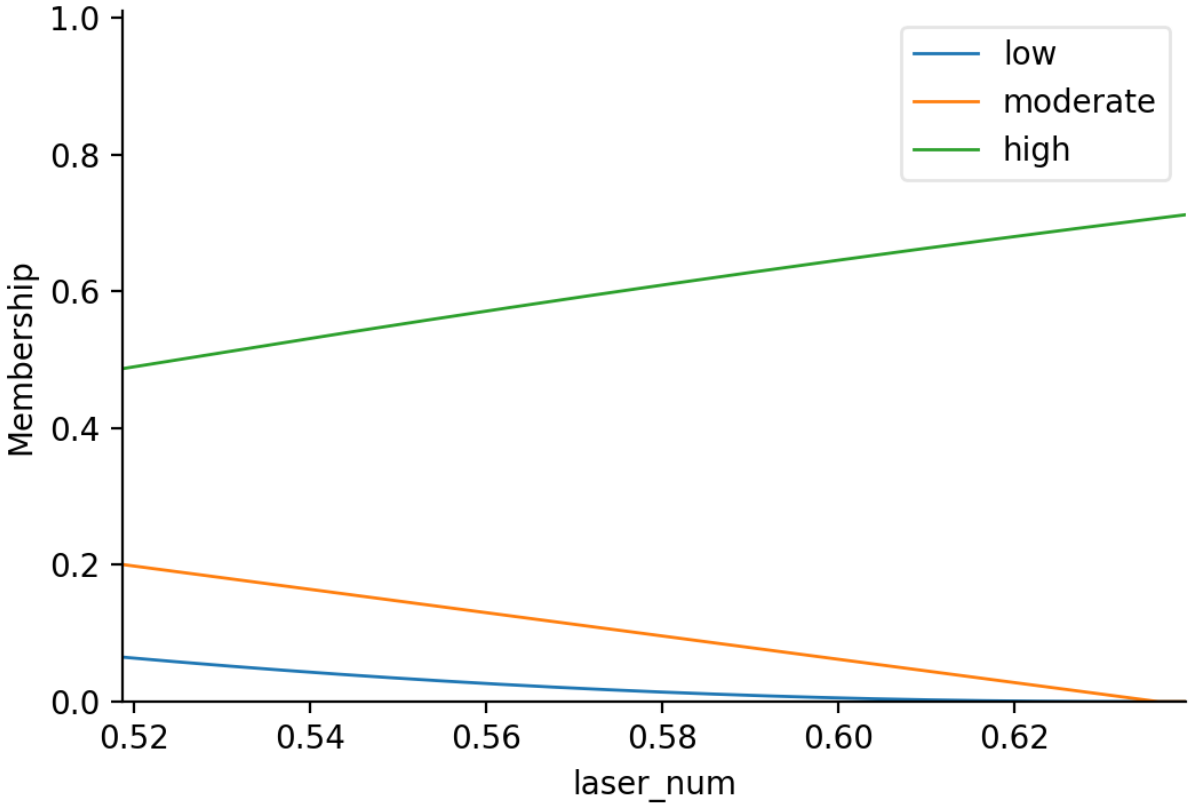



Figure A.23: Fuzzy membership of material descriptor: Laser Number in creep rate prediction.

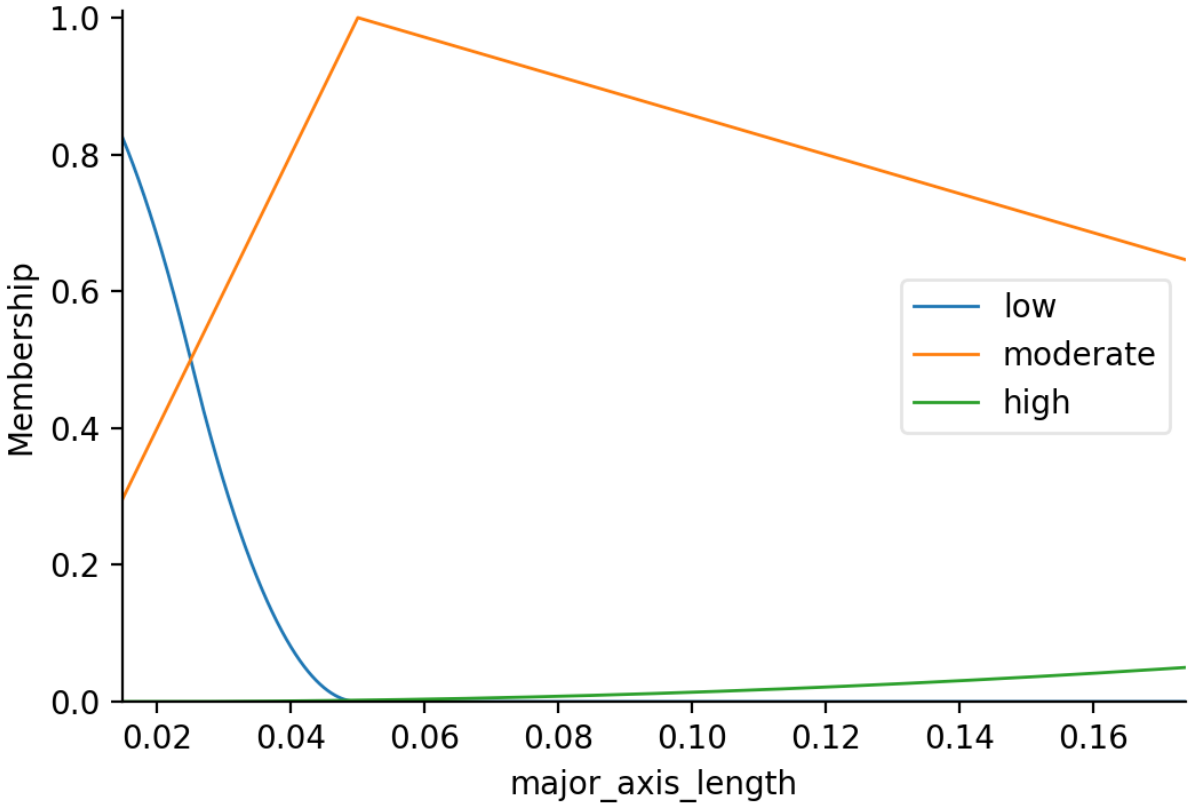



Figure A.24: Fuzzy membership of material descriptor: Major Axis Length in creep rate prediction.

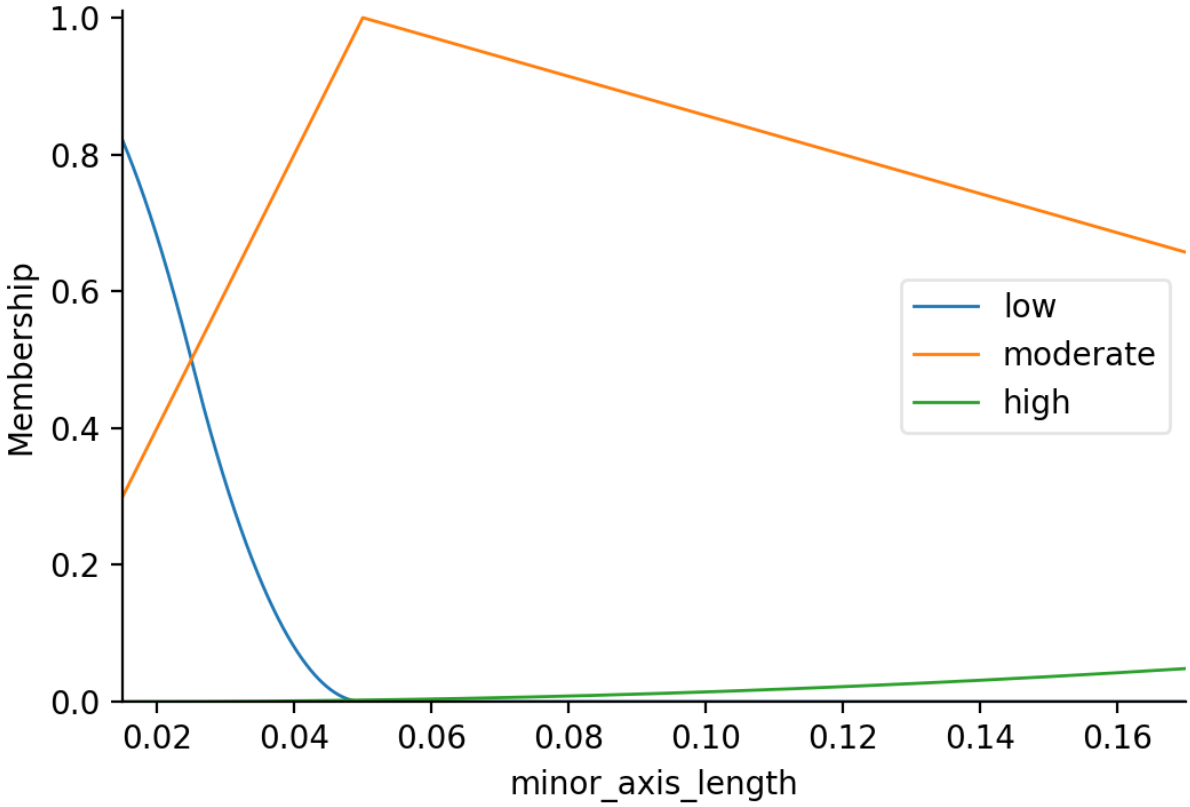



Figure A.25: Fuzzy membership of material descriptor: Minor Axis Length in creep rate prediction.

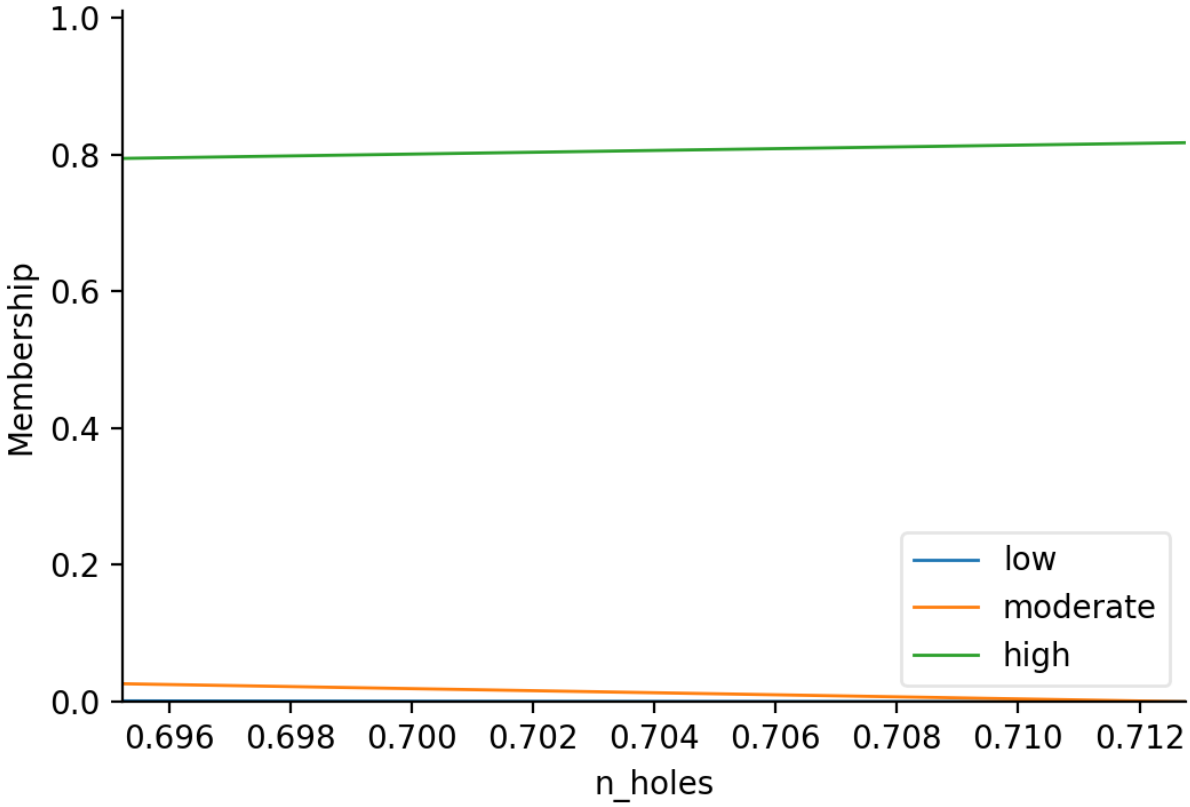



Figure A.26: Fuzzy membership of material descriptor: Number of Holes in creep rate prediction.

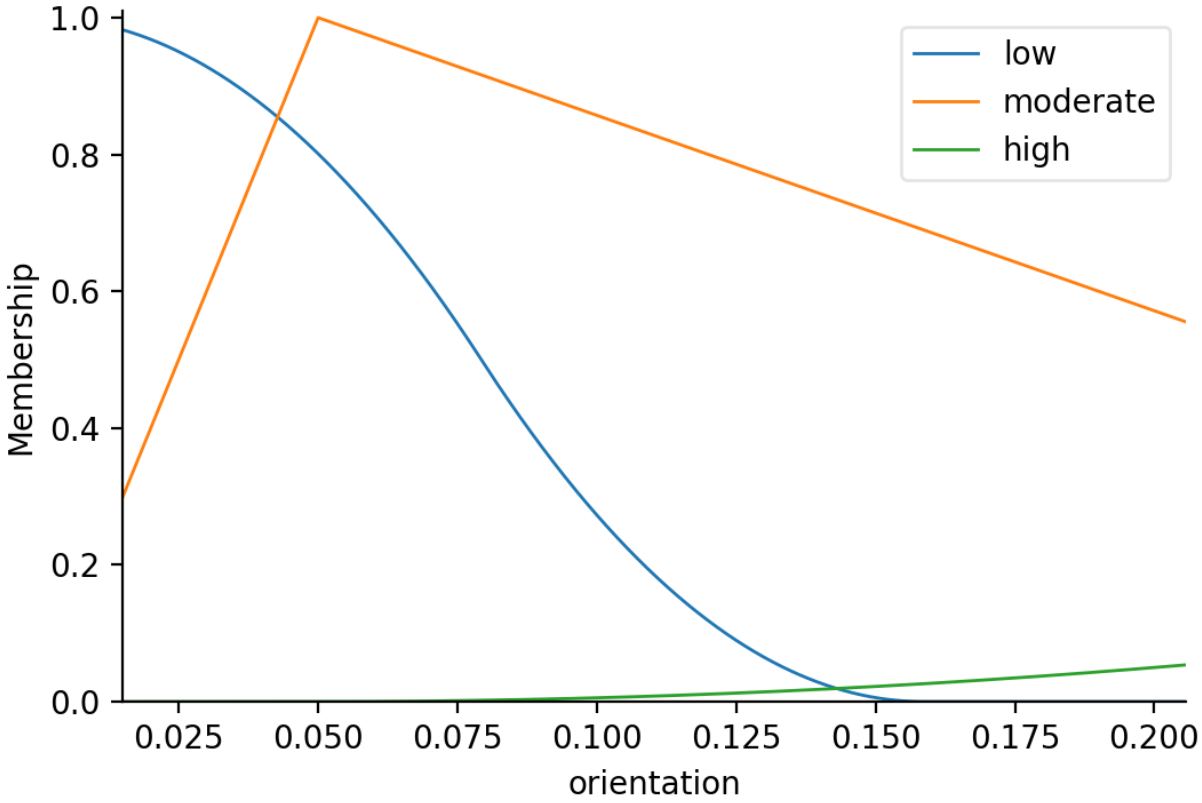



Figure A.27: Fuzzy membership of material descriptor: Orientation in creep rate prediction.

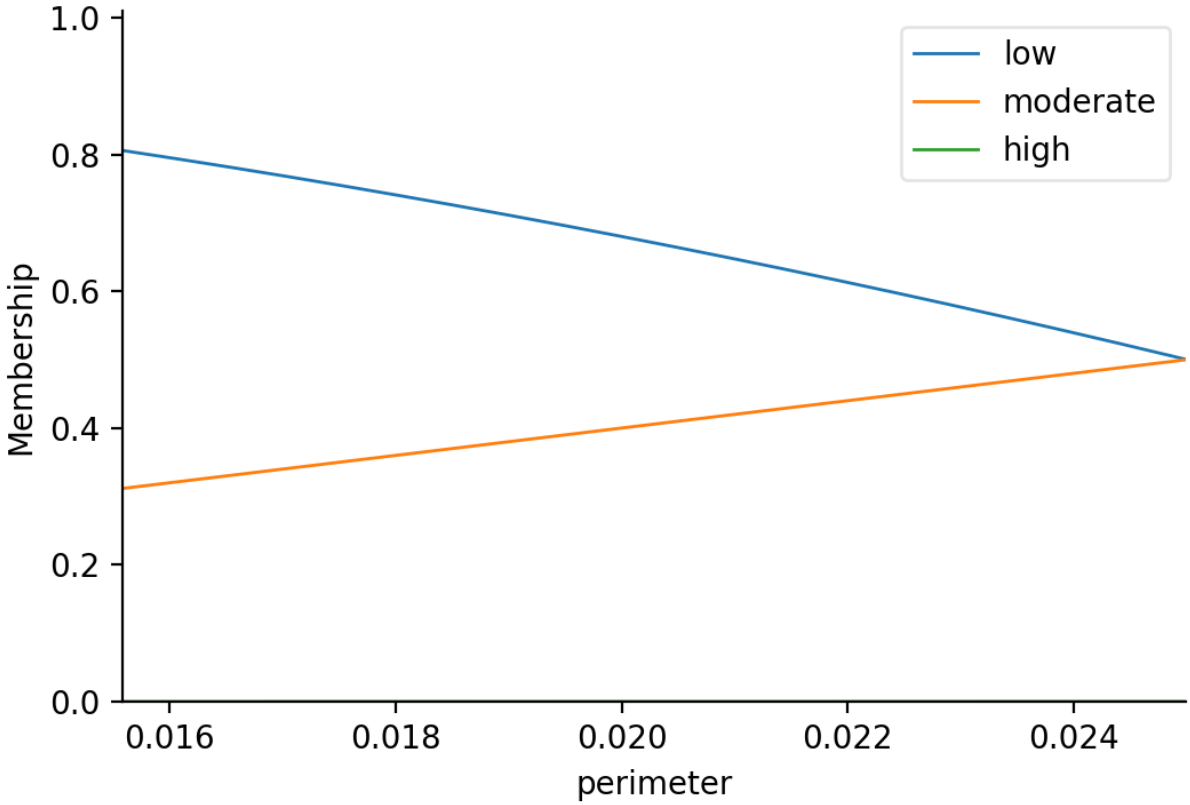



Figure A.28: Fuzzy membership of material descriptor: Perimeter in creep rate prediction.

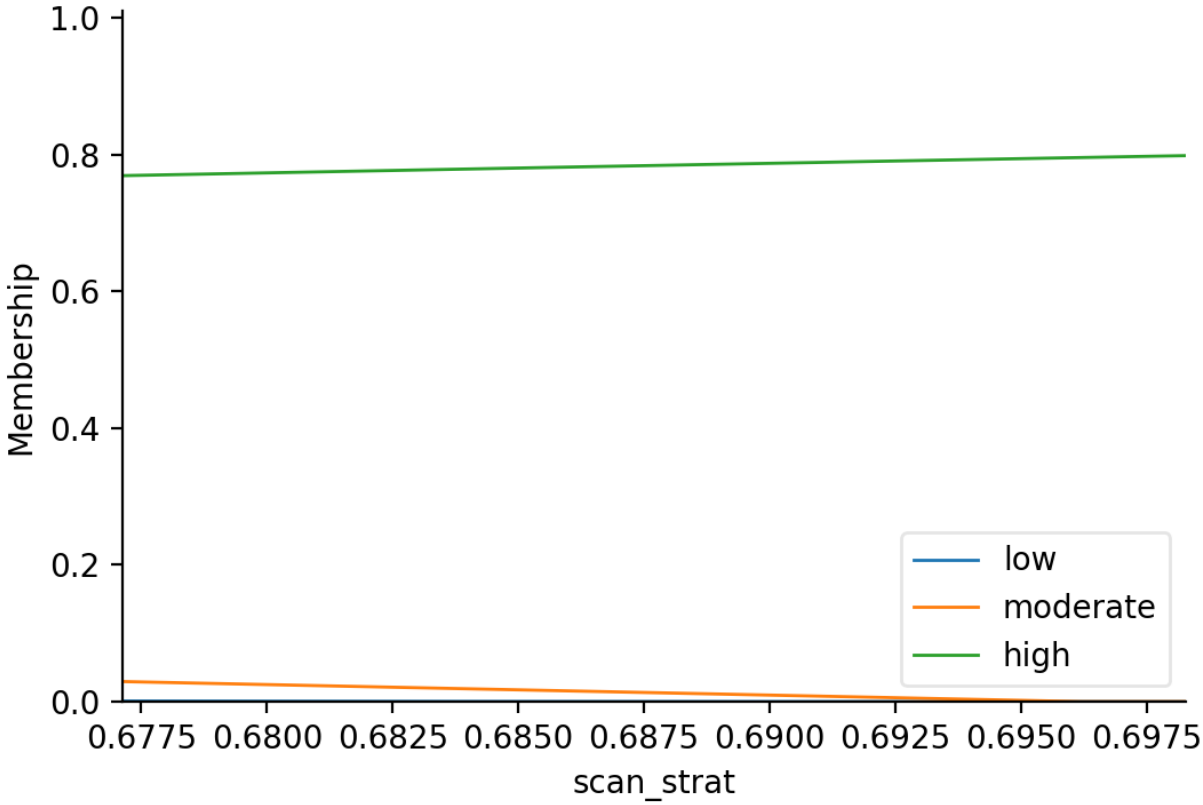



Figure A.29: Fuzzy membership of material descriptor: Scan Strategy in creep rate prediction.

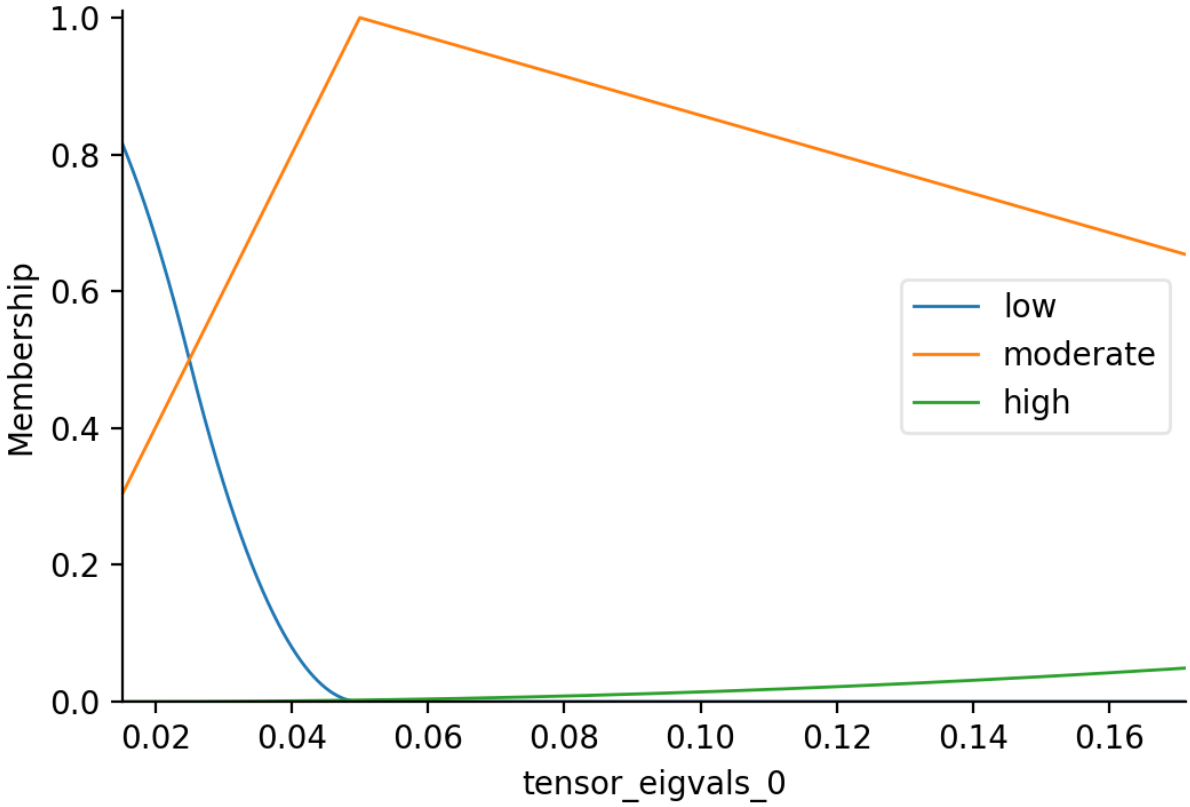



Figure A.30: Fuzzy membership of material descriptor: Tensor Eigenvalues 0 in creep rate prediction.

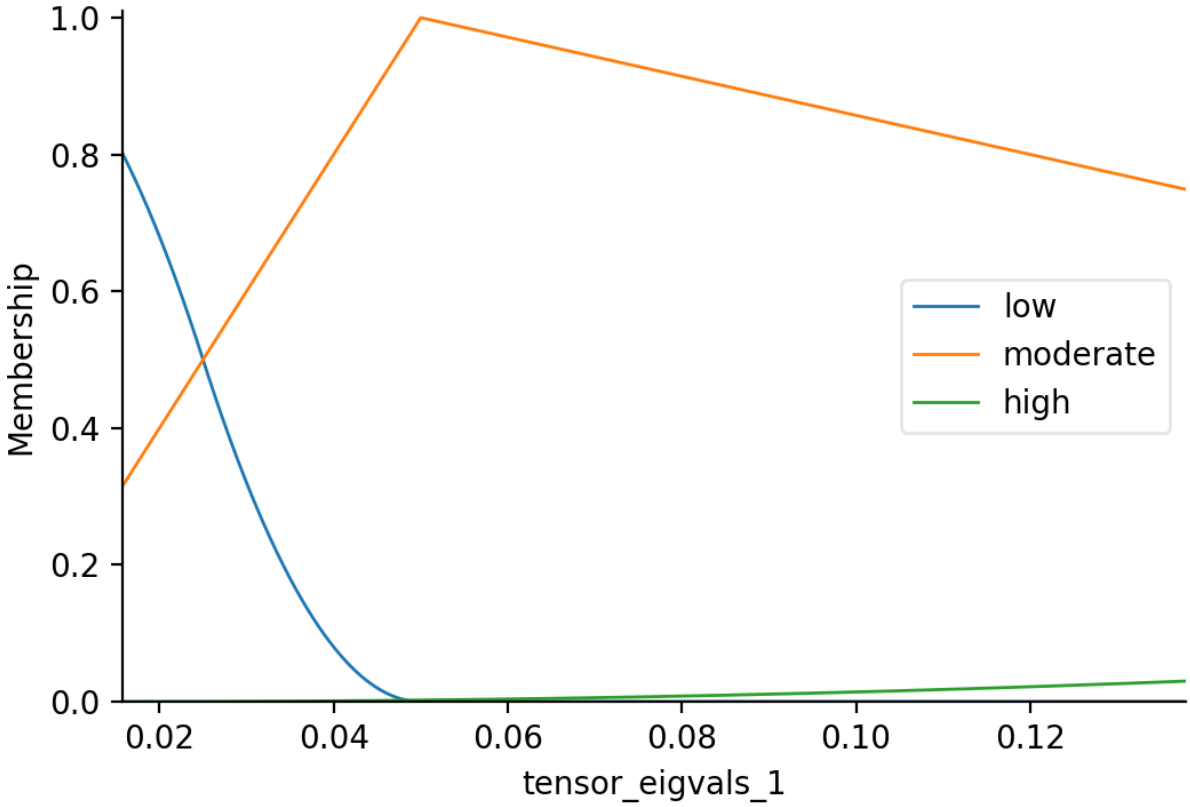



Figure A.31: Fuzzy membership of material descriptor: Tensor Eigenvalues 1 in creep rate prediction.